Online mapping and perception algorithms for multi-robot teams operating in urban environments

by

Johannes H. Strom

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
2015

Doctoral Committee:

Associate Professor Edwin Olson, Chair
Professor Edmund H. Durfee
Associate Professor Ryan M. Eustice
Associate Professor Ani Hsieh, Drexel University

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2015		**00-00-2015 to 00-00-2015**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Online Mapping and Perception Algorithms for Multi-robot Teams Operating in Urban Environments	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Michigan,Department of Computer Science and Engineering,Ann Arbor,MI,48109**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis investigates some of the sensing and perception challenges faced by multi-robot teams equipped with LIDAR and camera sensors. Multi-robot teams are ideal for deployment in large, real-world environments due to their ability to parallelize exploration, reconnaissance or mapping tasks. However, such domains also impose additional requirements, including the need for a) online algorithms (to eliminate stopping and waiting for processing to nish before proceeding) and b) scalability (to handle data from many robots distributed over a large area). These general requirements give rise to speci c algorithmic challenges, including 1) online maintenance of large, coherent maps covering the explored area, 2) online estimation of communication properties in the presence of buildings and other interfering structure, and 3) online fusion and segmentation of multiple sensors to aid in object detection. The contribution of this thesis is the introduction of novel approaches that leverage grid-maps and sparse multi-variate gaussian inference to augment the capability of multi-robot teams operating in urban, indoor-outdoor environments by improving the state of the art of map rasterization, signal strength prediction, colored point cloud segmentation, and reliable camera calibration. In particular, we introduce a map rasterization technique for large LIDAR-based occupancy grids that makes online updates possible when data is arriving from many robots at once. We also introduce new online techniques for robots to predict the signal strength to their teammates by combining LIDAR measurements with signal strength measurements from their radios. Processing fused LIDAR+camera point clouds is also important for many object-detection pipelines. We demonstrate a near linear-time online segmentation algorithm to this domain. However, maintaining the calibration of a eet of 14 robots made this approach di cult to employ in practice. Therefore we introduced a robust and repeatable camera calibration process that grounds the camera model uncertainty in pixel error, allowing the system to guide novices and experts alike to reliably produce accurate calibrations.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**108**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

ACKNOWLEDGEMENTS

Thanks to everyone who made this thesis possible, including my labmates during the MAGIC 2010 competition: Ryan Morton, Andrew Richardson, Pradeep Ranganathan, Robert Goeddel, and Mihai Bulic. Thanks to Chris Latham and the rest of the Australian DSTO and American TARDEC staff for organizing MAGIC, which provided funding and testing venues to support this work. Thanks also to my advisor Edwin Olson and my committee members Ryan Eustice, Edmund Durfee and Ani Hsieh for their time and encouragement. Finally thanks to my wife Kathryn whose support was invaluable, and without whom I certainly would not have made it through.

This work received financial support from a number of sources, including U.S. DoD grants W56HZV-04-2-0001, FA2386-11-1-4024 and Ford-UM Alliance award N015392.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS ii

LIST OF FIGURES v

LIST OF TABLES vi

ABSTRACT vii

CHAPTER

1. Introduction 1

1.1 Methods Overview 3

2. Map Rasterization 10

2.1 Problem Formulation 11
2.2 Method 13
2.3 Slice Caching 13
2.4 Spatially Non-Redundant Node Coverings 15
2.5 Transient Object Removal 16
2.6 Implementation 17
2.6.1 Evaluation 20
2.7 Discussion 24

3. Signal Strength Prediction for Teams of Robots 25

3.1 Background 27
3.2 Methods 30
3.2.1 Correlative Methods for Mobile Nodes 30
3.2.2 Multi-robot Tomography (MRT) 32
3.2.3 Multi-sensor ATTenuation Estimation (MATTE) . . 33
3.3 Evaluation 41
3.3.1 Test Apparatus 41

3.3.2 Results 44
3.4 Discussion 45

4. Colored Point Cloud Segmentation 46

4.1 Point Cloud Co-registration 49
4.2 Graph-based Segmentation 51
4.3 Experimental Setup 57
4.4 Results 58
4.5 Discussion 61

5. Guided Camera Calibration 63

5.1 Background 66
5.2 Method 68
5.2.1 Pixel-based Calibration Error Metric 69
5.3 Implementation 73
5.4 Human trials 74
5.5 Evaluation 76
5.5.1 Novice Calibrators using AprilCal and OpenCV . . 79
5.5.2 Evaluating the Max ERE metric 81
5.5.3 Accuracy of AprilCal on a Variety of Lenses 81
5.6 Discussion 83

6. Conclusion 85

BIBLIOGRAPHY 92

LIST OF FIGURES

Figure

1.1	Map rasterization	4
1.2	Robot communication	5
1.3	Scene segmentation	6
1.4	Camera calibration	8
2.1	Rasterized map of Keswick Barracks	11
2.2	Slam graph sequence from Keswick Barracks	12
2.3	Cache data structure	14
2.4	Phase 2 occupancy grid	19
2.5	Phase 2 cumulative runtimes	19
2.6	Keswick timing detail	21
2.7	Phase 2 timing detail	21
2.8	Phase 2 node detail	22
2.9	Map quality improvement from covering	23
3.1	Estimated attenuation map	26
3.2	Attenuation integral using the three-class representation	36
3.3	Five evaluation environments for MATTE	37
3.4	Evaluation of signal-strength prediction accuracy	42
4.1	Sample segmentation	47
4.2	Sensor configuration	50
4.3	Spatially-derived segmentation mesh	57
4.4	Segmentation evaluation	59
5.1	The AprilCal GUI	64
5.2	Max Expected Reprojection Error	70
5.3	Human study sample images	75
5.4	Human study error histograms for mean and max reprojection error	77
5.5	Human study focal- length and center distributions	78
5.6	Reprojection error heatmaps for AprilCal and OpenCV	80
5.7	Max ERE correlation with testing error	82
5.8	AprilCal evaluated on multiple lenses	82
5.9	Sample images for each lens used in the evaluation	83

LIST OF TABLES

Table

2.1	Rasterization methods used in the evaluation	18
2.2	Mapping datasets used in the evaluation of each rasterization method	18
3.1	Signal-strength prediction parameters	40
4.1	Colored point cloud segmentation parameters	58
5.1	Human study calibration performance evaluation	76

ABSTRACT

This thesis investigates some of the sensing and perception challenges faced by multi-robot teams equipped with LIDAR and camera sensors. Multi-robot teams are ideal for deployment in large, real-world environments due to their ability to parallelize exploration, reconnaissance or mapping tasks. However, such domains also impose additional requirements, including the need for a) online algorithms (to eliminate stopping and waiting for processing to finish before proceeding) and b) scalability (to handle data from many robots distributed over a large area). These general requirements give rise to specific algorithmic challenges, including 1) online maintenance of large, coherent maps covering the explored area, 2) online estimation of communication properties in the presence of buildings and other interfering structure, and 3) online fusion and segmentation of multiple sensors to aid in object detection.

The contribution of this thesis is the introduction of novel approaches that leverage grid-maps and sparse multi-variate gaussian inference to augment the capability of multi-robot teams operating in urban, indoor-outdoor environments by improving the state of the art of map rasterization, signal strength prediction, colored point cloud segmentation, and reliable camera calibration.

In particular, we introduce a map rasterization technique for large LIDAR-based occupancy grids that makes online updates possible when data is arriving from many robots at once. We also introduce new online techniques for robots to predict the signal strength to their teammates by combining LIDAR measurements with signal strength measurements from their radios. Processing fused LIDAR+camera point

clouds is also important for many object-detection pipelines. We demonstrate a near linear-time online segmentation algorithm to this domain. However, maintaining the calibration of a fleet of 14 robots made this approach difficult to employ in practice. Therefore we introduced a robust and repeatable camera calibration process that grounds the camera model uncertainty in pixel error, allowing the system to guide novices and experts alike to reliably produce accurate calibrations.

CHAPTER 1

Introduction

Cooperative multi-robot teams have the potential to replace humans in tasks such as search and rescue, urban reconnaissance, and surveying. For example, robots could be used for identifying and aiding victims in a natural disaster, searching for roadside bombs, finding lost items, tracking intruders, or building inspection [1, 2, 3, 4]. These tasks are often difficult, dangerous or unpleasant, so if robots can be deployed instead of humans then society may benefit. For instance, although robots were only a small fraction of the agents involved in the cleanup at Fukushima, they reduced the need to expose humans to harmful radiation [5]. Such tasks are often inherently parallelizable, which means many robots can cooperate to complete them faster. For example, when searching for victims, multiple robots can split up a search area, completing the search faster than a single agent could.

Despite the potential benefits of multi-robot teams to such applications, there has yet to be a sustained deployment of automated multi-robot systems to real-world environments. The dominant paradigm remains human-teleoperated robots (e.g. explosive ordinance robots) or carefully engineered environments (e.g. KIVA's robotic warehouses) [6, 7]. Excepting cost, reliability and mechanical design (which are not to be overlooked), the main remaining challenges preventing deployment of such systems are the lack of online and scalable autonomy systems to control many

robots with little or no human intervention.

In this thesis, we focus on a subset of these algorithmic and logistical challenges, particularly in mapping and perception, which are exacerbated when multiple robots are deployed cooperatively. Algorithms for field robotics should generally be *online* and *scalable*. By online, we mean that an algorithm should be capable of running while the robot is in operation, without falling behind. Typically this means runtimes bounded by a second or two, but ideally much less. For example, an online motion planning algorithm would not require the robot to stop and think before deciding what the next action is. By scalable (in this context), we mean that the algorithm remains online, while accommodating data arriving from a dozen or more robots at once. For example, a non-scalable mapping algorithm may have combinatorial complexity with respect to the size of the environment and number of robots. In practice, many mapping and perception algorithms, such as FastSLAM or Fisher image classification, struggle to scale to large numbers of robots or to run online, respectively [8, 9].

The work in this thesis is also motivated by the challenges we faced when the APRIL lab competed in the MAGIC 2010 multi-robot reconnaissance competition, where teams of autonomous robots were tasked to explore a 500×500m indoor-outdoor urban environment, and identify and neutralize simulated bombs [10]. In particular, it highlighted the shortcomings of existing work in the key areas of 1) online map rasterization from multiple robots in a large area, 2) online estimation of communication properties, and 3) online sensor fusion and scene segmentation for colored point clouds. This thesis makes improvements over the state of the art in each of these areas with the goal of improving the competency of multi-robot teams in MAGIC-like environments (among others). While these topics share a common application (multi-robot teams), there are also many similarities in the technical approaches we used to overcome them (multi-variate Gaussian inference and grid-based spatial representation). While MAGIC 2010 is a key motivator for the work presented here,

this thesis contributes to the broader robotics community, where large-scale mapping, state estimation and perception are topics of ongoing research. Improving the state of the art in the aforementioned areas will help others in this field, for example by reinforcing how spatial reasoning with rasterized gridmaps can be coupled with parametric state estimation techniques to expand the range of applications where robots can be deployed autonomously.

1.1 Methods Overview

A key challenge for multi-robot teams is generating a fused state estimate that incorporates sensor data from all robots in the team. In particular, multi-robot frontier-based exploration requires a live, coherent estimate of which part of the environment is already explored, and which is still unexplored [11]. The border between these two regions is called the 'frontier'. In order for online planning to occur, this state estimate must be available with limited delay (e.g. < 2 seconds), or else robots must stop and wait until a new command is available (e.g. from the centralized planner).

Fortunately, existing GraphSLAM inference techniques, in combination with efficient loop closure generation, extend naturally to the multi-robot case [12, 13, 14]. However, these estimation methods provide only the maximum-likelihood trajectories of the robot team. The configuration of the environment, such as the position of walls and doors, must still be computed explicitly from the accumulated sensor data, for example by rasterizing the position of obstacles into a grid-based representation of the environment. Since the estimates of where a robot has traveled are constantly updated as new loop closures are identified, naïve approaches to this problem, which re-rasterize the entire world after each update, cease being "online" even for modestly sized environments.

To enable online planning for the team of robots, Chapter 2 describes a new two-

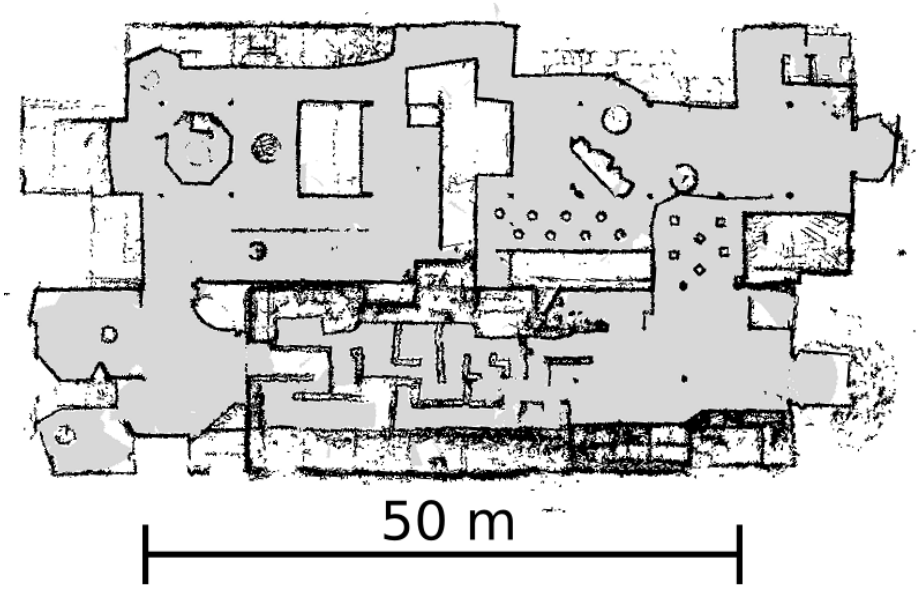


Figure 1.1: Map rasterization. Rasterizing a map of the environment allows robots to understand where they can drive and where the walls are.

level rasterization technique which scales better to larger environments with many robots. In particular, on our largest dataset with 14 robots covering 220x160m, the naïve algorithm consistently took over 6 seconds to rasterize all the data, while the proposed method averaged less than 1 second per rasterization.

Having a consistent, low-latency rasterized map of the environment is a necessary prerequisite to enable frontier planning approaches, however rasterized maps are not sufficient to enable the success of such algorithms. Multi-robot teams also generally need to communicate in order to collaborate effectively, for example to share sensor data, or agree on task allocations. With our MAGIC 2010 system, we deployed a long-range centralized radio infrastructure that struggled to perform well in dense urban environments. During the MAGIC competition the system averaged only 108 bytes/second, shared between 14 robots. To put that in perspective, our heavily-compressed sensor scans average 132 bytes [2]. The lack of bandwidth with this system was a key limitation in scaling our system to larger environments, or more robots. A common approach to improving bandwidth is to use shorter-range, ad-

Figure 1.2: Robot communication. When operating in urban environments, robots need to understand where they will be able to communicate in order to collaborate effectively.

hoc mesh networks. However, to prevent a robot from becoming detached from the communication graph, robots need to plan to remain in contact, which requires predicting where they will be able to communicate. Unfortunately, signal strength prediction is very challenging. Even when prediction is confined to two points on a 2D plane, the estimation problem is fundamentally under-constrained, as the input space is $\mathcal{R}^4$, and there are very few labeled outputs because only the signal strength between each robot is known.

Chapter 3 describes a new method for predicting signal strength that leverages the existing capability to generate LIDAR-based maps in combination with the radio's signal strength estimates to provide stronger and more informative constraints on the estimation problem. We then formulate a multi-variate Gaussian estimation problem, embedded in a grid that encompasses the workspace, whose maximum likelihood (ML) solution estimates the signal attenuation in each grid cell. The resulting signal

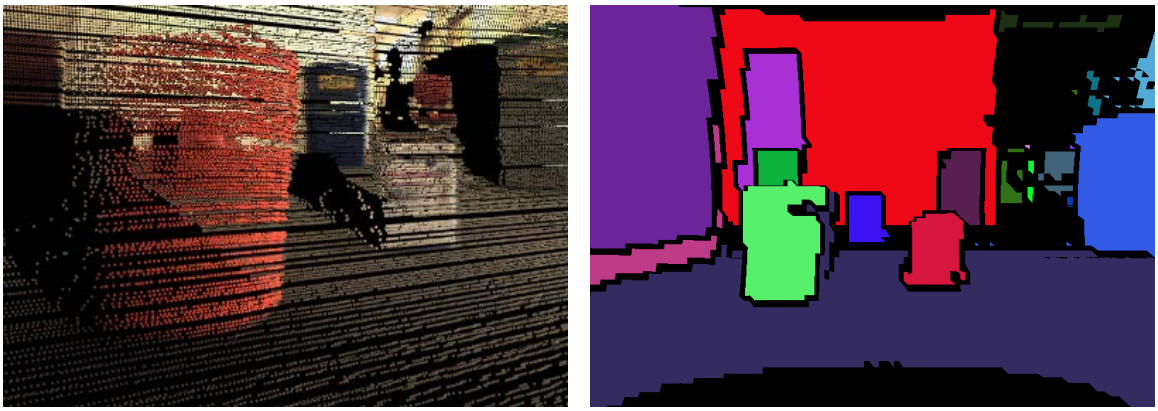


Figure 1.3: Scene segmentation. Efficient segmentation of colored point clouds (left) is important to enable robots to detection recognize objects in real environments (right).

attenuation maps (which can be used to predict signal strength using a line-integral), can then be used as input to a communication-aware planner to dispatch robots such that connectivity is maintained [15].

In addition to collaboratively mapping the test environment, the MAGIC 2010 competition also required identification of simulated improvised explosive devices. These objects had a particular shape and color which robots were required to identify and avoid, otherwise the robot would be deactivated if it came too close. Both 3D LIDAR and color information are required for accurate classification. Furthermore, robots must be able to quickly (online) classify their surroundings in order to avoid running into one of these hazards.

To tackle this challenge, in a joint project with Andrew Richardson, we adapted a popular graph-based, image-only segmentation technique to the colored point-cloud domain. The adaptation, presented in Chapter 4, is not straight-forward because color and range measurements are not directly comparable, and furthermore they have different sample densities. The resulting algorithm was able to run at approximately 4Hz on a single core, which meets our "online" requirement. Unfortunately, our proposed algorithm relies on carefully co-calibrated LIDAR and camera sensors which made it logistically prohibitive to implement the proposed algorithm with our 14-robot

team using the error-prone camera calibration toolboxes available at the time.

The final contribution of the thesis, also joint work with Andrew Richardson, improved the state of the art in camera calibration to be more robust and repeatable than state-of-the-art methods, even for novice users. Most existing camera toolboxes require an expert to select several images of a calibration target to compute the many intrinsic parameters, such as focal length and radial distortion. However, the resulting parameter estimates are sensitive to the chosen calibration images (the training set). It can be difficult, even for experts, to ensure that enough images are incorporated to correctly estimate all parameters. Principled validation of a calibration can theoretically be done by collecting a large, exhaustive and independent test set, but this time-consuming process is not widely adopted in practice [16].

In contrast to the expert-driven approach, AprilCal, described in Chapter 5, introduces a principled, algorithm-driven process that suggests calibration target positions in order to minimize a new metric called "Max Expected Reprojection Error" (Max ERE). This metric evaluates the quality of an in-progress calibration based on the expected error throughout the image, rather than only evaluating on the user-chosen training set, as in previous approaches. We demonstrate that the Max ERE correlates highly with empirical test set errors, eliminating the need for end users to spend the time to collect their own. Principled computation of the Max ERE is made possible by modeling the camera intrinsics and calibration-target extrinsics parameters as a sparse, multi-variate gaussian distribution. This allows both efficient inference, using similar tools as in the map rasterization and signal strength estimation works, but also for straight-forward marginal computation. The Max ERE is then derived by sampling over a particular marginal distribution. The resulting interactive system allowed users to reliably calibrate a variety of camera lenses. We validated this claim with a user study that showed that novice users with no prior experience could reliably calibrate cameras to $<<$ 1 average pixel error. Because this system provides

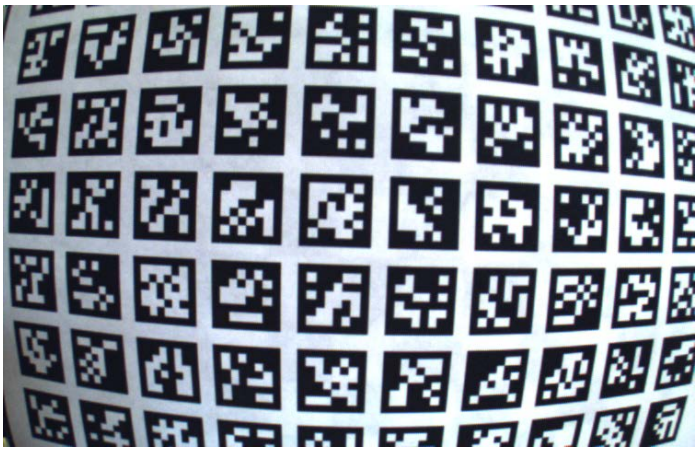


Figure 1.4: Camera calibration. Human-selection of target images (especially by novices) results in inaccurate parameter estimates. Our guided, repeatable calibration process helps make calibration of a fleet of robots manageable.

live suggestions to the user, it must achieve online performance.

These novel contributions to map rasterization, signal strength estimation, point cloud segmentation and camera calibration have the potential to improve the capabilities of cooperative multi-robot teams. Many of these contributions use similar spatial- and probabilistic-reasoning tools to deliver online algorithms applicable to real robots. In particular, the contributions of this thesis are:

- A new, more efficient algorithm for map rasterization that enables multiple simultaneously exploring robots to explore large urban environments
- A novel approach to, and online algorithm for, signal-strength prediction that handles teams of robots
- An efficient, online, colored point cloud segmentation algorithm that improves the ability to segment objects from their surroundings
- A principled method for reasoning about calibration uncertainty, which enabled the creation of the first intelligent, guided camera calibration toolbox

These contributions are presented in Chapter 2 “Map Rasterization”, Chapter 3 “Sig-

nal Strength Prediction for Teams of Robots" Chapter 4 "Colored Point Cloud Segmentation" and Chapter 5 "Guided Camera Calibration", respectively.

CHAPTER 2

Map Rasterization[1]

Successful autonomous operation of a team of robots in unknown large-scale environments depends on having a robust and globally consistent mapping solution. In particular, rasterized occupancy grids have many important applications where efficiency is an important objective (see Fig. 2.1). For example, autonomous exploration presents particularly hard timing requirements — the planner requires map updates quickly so that it can compute updated plans without requiring robots to stop and wait [17].

While parametric SLAM graphs can support efficient maximum-likelihood inference for very large graphs, (e.g., using iSAM [13]), this graph-based representation is not directly useful to frontier-based exploration algorithms, which require a globally-consistent view of the border (frontier) between explored and unexplored space. Standard approaches to rasterization in the Simultaneous Localization and Mapping (SLAM) field scale linearly with the amount of robot sensor data, eventually overwhelming a fixed CPU budget. We present a method that uses a sparse parametric SLAM algorithm to enable occupancy grid updates to be computed online for larger environments than any naïve methods can handle. While our approach still has a linear worst case, it is designed to produce much faster performance in

[1] ©2011 IEEE. Adapted with permission, from Johannes Strom, Edwin Olson, "Occupancy Grid Rasterization in Large Environments for Teams of Robots" September 2011.

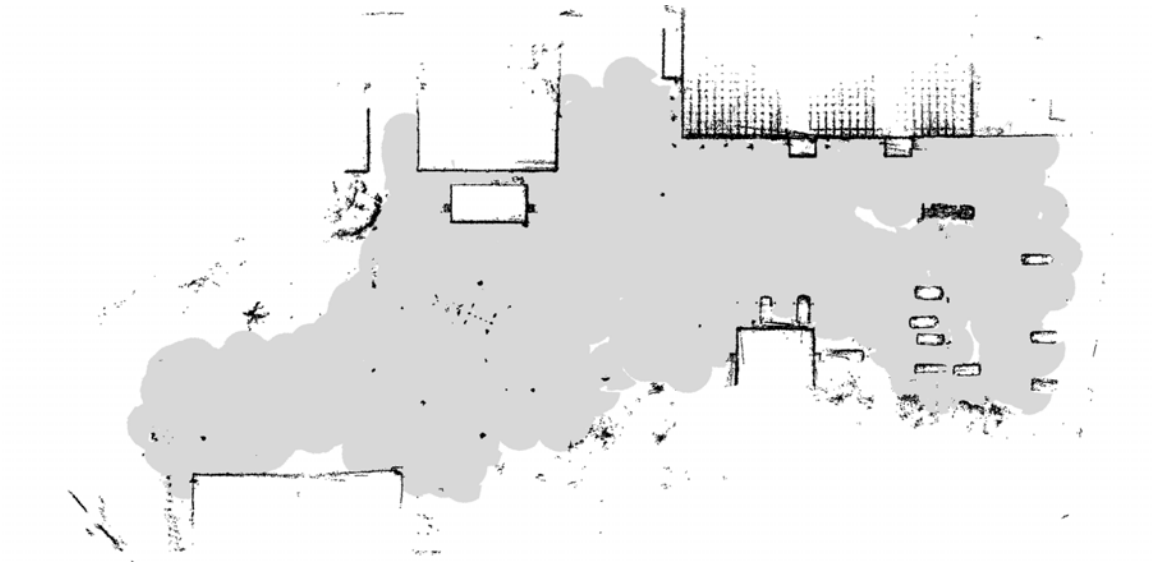


Figure 2.1: Rasterized map of Keswick Barracks. The 180 x 100 m map was produced with a mean processing time of 1.3 seconds on a single 2.53 GHz core. Permanent structure is shown in black; known free space in gray; unobserved in white.

typical urban environments. Our method exploits several properties of the mapping problem: We observe that after incorporating loop closures, the posterior, maximum-likelihood (ML) estimate of a historical robot position does not necessarily change, so parts of the gridmap may remain the same. We also observe that useful maps, such as for cooperative planning, only contain the static structure of the environment, omitting dynamic obstacles. Finally, we note that with some low-noise sensor types, such as laser range finders, a small number of views of a place are sufficient to form a complete map. These observations allow us to design a new data structure and principled data reduction technique which enabled online performance in domains where naïve algorithms were not able to keep up.

2.1 Problem Formulation

We consider the problem of generating a rasterized map using the laser-range-finder data associated with the trajectories of a team of robots. Each map consists of a grid of ternary labels *free*, *structure*, and *unknown*. In particular, we build on

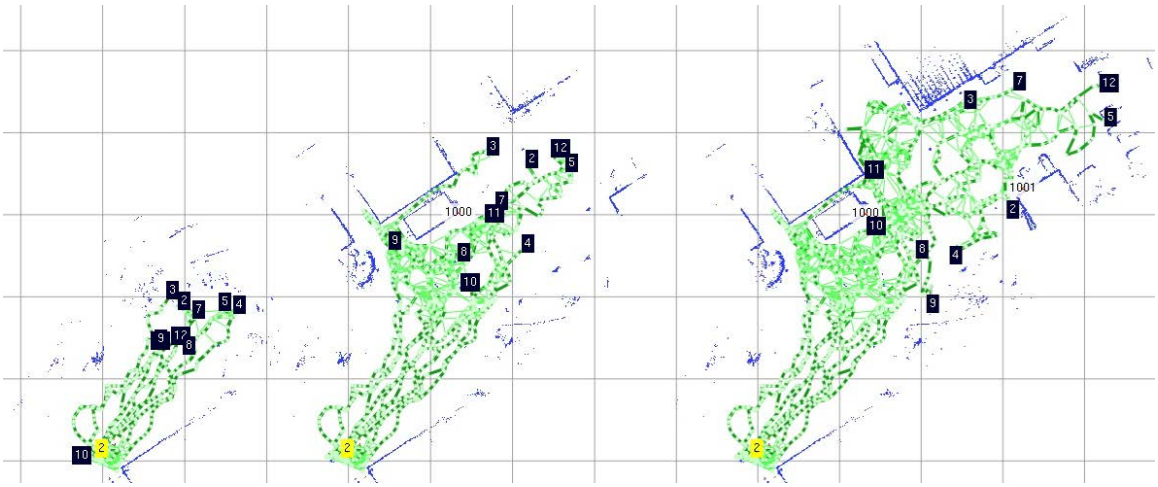


Figure 2.2: Slam graph sequence from Keswick Barracks. Even though the SLAM graphs do not directly encode a grid map, we've shown that we can use this representation to efficiently compute an occupancy grid of the observed space.

the GraphSLAM formulation, where a SLAM graph G contains a set of nodes V and edges E. The nodes represent the location of a specific robot at a specific instant in time. Each node is associated with a local occupancy grid taken at that moment. The edges represent constraints between two poses that determine their relative position, including a covariance. An edge can be created using odometery and IMU readings from a single robot to form that robot's trajectory. Furthermore, additional edges can be added by finding correspondences in the sensor data associated with two nodes. Several efficient methods exist for computing ML posteriors for the node locations, such as SqrtSAM, iSAM, or SGD [13, 18, 19].

Given the sensor data associated with each node and its ML posterior position, we can formulate a basic naïve algorithm for building a global occupancy grid map. Starting with an empty map, we iterate over each node in the graph and rasterize the associated sensor data onto a global map. This rasterization algorithm scales linearly in the number of nodes in the graph: for applications where only the final map is desired, this method works quite well.

In the case of a team of actively exploring robots, we actually operate on a *sequence* of n graphs $G_1, \cdots, G_n$, from which we want to produce maps $M_1 \cdots M_n$, e.g, for use

with an autonomous planner. We desire that the maps be computed on-line with a minimum of delay so that consumption by the planner can proceed directly. We further require that the maps reflect the exact Maximum Likelihood Estimate (MLE) position of all the sensor data contained within it, and that moving obstacles are removed. (Dynamic obstacle avoidance is not handled at the task allocation level during frontier planning, but rather using lower latency, on-board collision avoidance.)

2.2 Method

We use a combination of techniques to speed up the computation of M_{i+1} given previous maps $M_1 \cdots M_i$. We first introduce a spatially-aware data structure that uses caching to automatically exploit the structural similarity between subsequent maps. This is possible because we are primarily concerned with mapping the structure of the environment, which is assumed to be static during the course of a mission. Second, we introduce the notion of a "node covering" that allows M_i to be computed by discarding redundant sensor data. These optimizations enable the time to compute a rasterization to scale with the size of the observed area, not with the duration of the system's operation (which could be quite long, or even unknown). Finally we show how this method can incorporate transient object removal.

2.3 Slice Caching

Each subsequent node set is a superset of the previous set ($V_i \subseteq V_{i+1}$), so we could theoretically improve on the naïve algorithm if we can exploit the overlap. For example, if the edges added to G_i to form G_{i+1} did not include significant information gain (in the probabilistic sense), then the maps M_i and M_{i+1} can be quite similar. What we desire is a principled way to determine which portions of M_i we can reuse to form M_{i+1}, so that if the graph is not gaining information, the map update is very

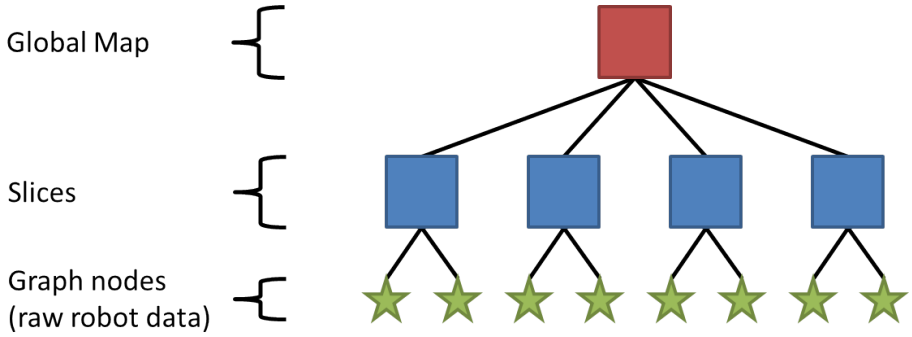


Figure 2.3: Cache data structure. The leaves, depicted here for $N = 2$, represent local maps from each robot. A single layer of globally aligned sub-maps are stored in the slices.

fast. In cases where the graph changes significantly, we must be willing to wait for an accurate map M_{i+1} to be formed if we want to ensure the map is consistent.

We can exploit similarities in subsequent maps by partitioning the nodes V_i into k disjoint subsets $S_1 \cdots S_k$ which we call *slices* (see Figure 2.3). A system designer can choose k freely, to tradeoff between faster best-case performance, and faster average-case performance, as discussed in the evaluation. We then compute small maps m_j from each slice S_j, such that the final map M_j is the rasterization of all $m_1 \cdots m_k$. For each map update, the number of subsets that need to be converted to sub-maps will vary. In the worst case all subsets will contain nodes whose ML posterior has changed and the data from every node will be examined twice — once to form m_j, and once again to form M_i. To achieve faster map updates, we can avoiding recomputing m_j if the ML posterior position of no node in S_j changes enough to cause a change in m_j. Determining whether to recompute m_j can be done exactly by observing whether a change in the ML position would result in more than 1 grid cell of change in its projection into the global frame. As we show in our evaluation, it is usually the case that enough nodes are *not* part of a loop closure that it is worthwhile to cache m_i.

The method for partitioning V_i into subsets greatly impacts the runtime of the algorithm. Unfortunately it is not possible to know an optimal partitioning in advance. Ideally, we design the subsets to contain highly correlated graph nodes. This

way, if l nodes need to be recomputed, only $\lceil \frac{|S_k|}{l} \rceil$ slices need to be recomputed. If the subsets were not correlated, we might need to recompute as many as l slices. We found that in practice, a good heuristic partitioning is to automatically group N sequential nodes from a single robot into a slice. These slices contain spatially and causally correlated nodes, so their ML posteriors are likely to change together. This means a single node is less likely to cause the recomputation of an entire slice unless the other nodes in that slice also have moved.

2.4 Spatially Non-Redundant Node Coverings

To further increase the speed of computing the rasterization for a map, we note that much of the sensor data collected by the robots is redundant. For environments that are structurally static, such as indoor-outdoor urban environments, a small number of sensor observations can be sufficient to build an accurate map of a small area. In particular, this is the case for low-noise sensors such as laser range finders.

This observation allows us to reformulate the occupancy map generation using only a subset of nodes, $C_i \subseteq V_i$, which *cover* all the nodes in N_i. We define a set of nodes to be covering if there exists no node not in C_i that adds significant information to the map M_i. We can also show that this assumption actually allows us to improve map quality subjectively, since multiple observations of a single place can only result in blurring of features, potentially erasing important topological features such as doorways, which can result in an incomplete exploration mission.

In practice, determining a minimal covering that meets this definition is at least as expensive as computing M_i' from V_i, since it is unclear without examining each local map m_j in detail if it will contribute significantly to the global map. However, we can define a conservative heuristic that will guarantee a complete covering in most environments, though it may not be minimal. We solve this by automatically considering the field of view that each local map covers, and the location at which

it was taken. If both of these are too similar to an existing measurement, we can remove it from the covering.

The covering paradigm enables a smart decimation of the sensor data, which enables the computation of M_i to be bound by the area it covers, rather than by the time the system has been running. In the worst case, the area could still be linearly dependent on the runtime. However, at system design time, we will likely have a prior over the size of the environments we are mapping, even if we don't have a prior over its runtime.

2.5 Transient Object Removal

The final step of the rasterization process is determining how to ensure transient objects are not included in the final map. As previously mentioned, transient objects cause problems for autonomous planners because their presence can obscure the existence of more explorable terrain through a doorway. As a result, we must carefully detect which sensor readings are due to transient objects. In principle, such objects are easy to detect by looking for disagreement among sensor data as to the free or occupied status of a cell. In practice however, some disagreement can be caused by alignment error and does not necessarily indicate a transient object. In cases where alignment is poor (e.g., when the SLAM Graph is poorly constrained), we want to mitigate the possibility that entire walls will be removed. The "bias" to label an object as *transient* reflects the degree to which the system designer wishes to be resistant to alignment error versus the desire to ensure all doorways are correctly marked as passable terrain. An efficient solution to this problem is to view multiple sensor readings of a particular cell as observations of a binary variable with equal covariance.

The probability of the cell being an obstacle is given by:

$$p(obs|z_0 \cdots z_n) \sim p(obs) \prod_i p(z_i|obs)$$

This can be computed directly by summation of log probabilities [20]. A simple threshold on $p(obs|z_0 \cdots z_n)$ is then used to determine whether the cell should be marked as occupied or free, reducing the process to a vote at each cell. Note that while the covering technique presented in the last section reduces the number of sensor observations used, the covering algorithm is designed to preserve sufficient overlap so that dynamic objects are still be detected.

Algorithm 2.1 COVER_COMPOSITE($G_i = \langle V_i, E_i \rangle$)

```
1: for v ∈ V_i | v ∉ V_{i-1} do
2:    r = robot(v)
3:    if coverTest(v, C_r) then
4:       append(C_r, v)
5:       s_k = getNextSlice(S_r)
6:       addToSlice(s_k, v)
7:    end if
8: end for
9: for s_j ∈ S do
10:   if projectionError(s_j) >thresh then
11:      regenerate(s_j)
12:   end if
13: end for
14: M_i = initialize(boundingBox(S))
15: for s_j ∈ S do
16:   compositeOnto(M_i, s_j)
17: end for
18: threshold(M_i)
19: return M_i
```

2.6 Implementation

Our method, '*CoverComposite*', which combines the improvements from the node covering with slice caching, is described in Algorithm 2.1. We assume that another

module, such as GraphSLAM, is producing a sequence of ML posterior SLAM graphs $G_i, G_{i+1}, \cdots$. The algorithm we present is run on successive G_i. For convenience, the cover and slice sets are stored in memory as C_r and a set of slices S_r for each robot $r \in \{0, \cdots, R\}$. First, all the new nodes that first appear in graph i are examined to determine if they belong in the covering for the robots which spawned them. If so, they will be added to an existing slice if there is room, or otherwise force the creation of a new slice. Next, each slice is examined to determine if the geometric projection error of any update of the state of any of its member nodes would cause an error of more than some threshold (e.g., 1 pixel). If an update is needed, the cumulative gridmap for that slice is regenerated according to the most recent ML posteriors for each node. Finally, all the slices for all robots are composited together to form the final occupancy grid. We also track of how many slices observed each cell as either observed or free. Using the probabilistic scheme described in the previous section, we convert that representation to a binary map via a fixed classification threshold.

The asymptotic complexity bounds of our method are the same as the naïve algorithm we described earlier. In the worst case, our implementation is at most twice as slow, since each sensor datum must be examined at most twice. However, we have shown empirically that the average run time is much faster than the naïve rasterization algorithm.

Algorithm	**Nodes composited**
NaïveComposite	All
NaïveCoverComposite	Cover (every timestep)
CoverCompositeN	Cover (only when necessary)

Table 2.1: Rasterization methods used in the evaluation

Dataset	**Size**	**# robots**	**Num. Graphs**
Keswick Barracks	150 x 150 m	10	68
Showgrounds (phase2)	220 x 160 m	14	429

Table 2.2: Mapping datasets used in the evaluation of each rasterization method

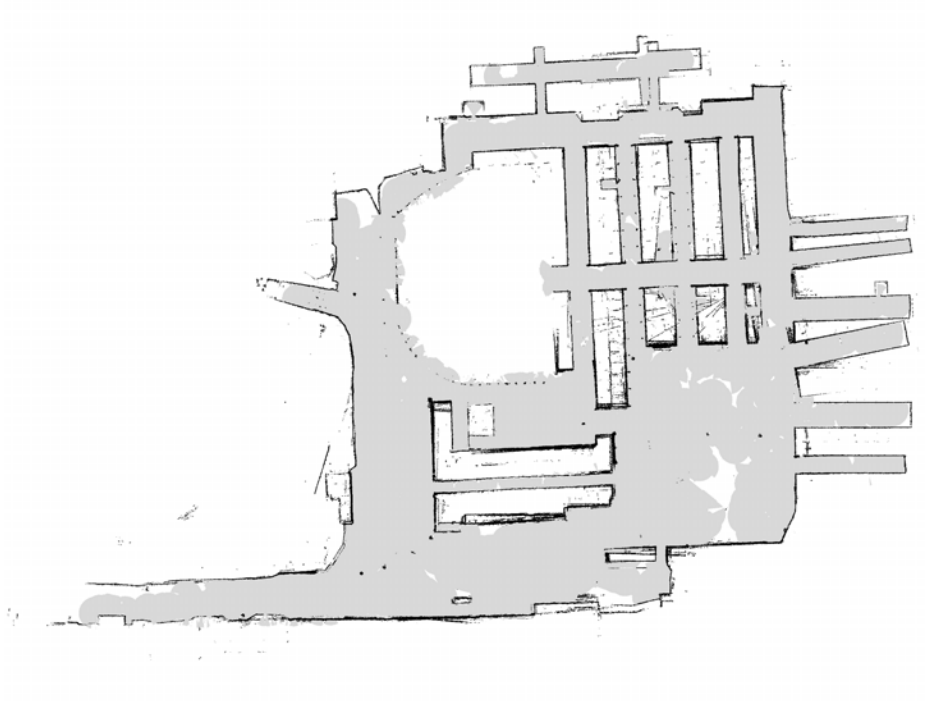


Figure 2.4: Phase 2 occupancy grid. This grid map includes data from 15 robots collected over the course of 72 minutes. Size: 220 x 160 m @ 0.1m resolution.

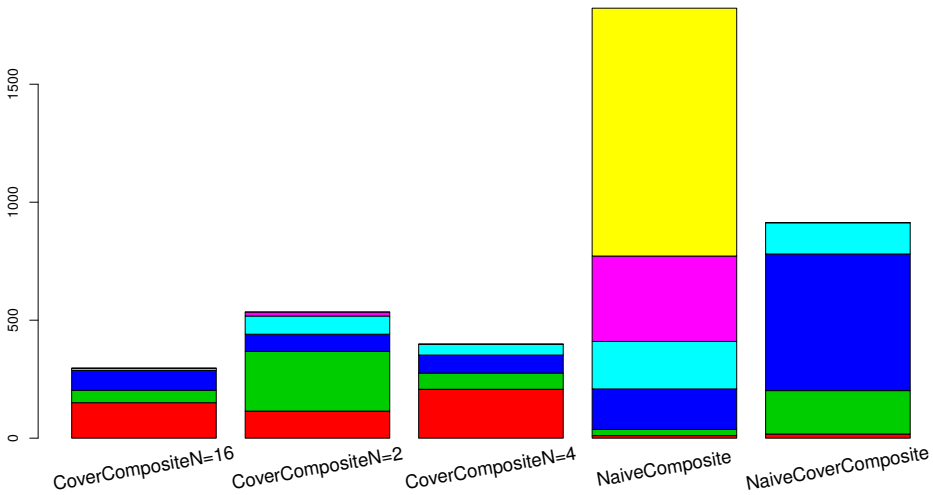


Figure 2.5: Phase 2 cumulative runtimes. Color bands indicate the portion of rasterizations that finished in <1.0s, <2.0s, <3.0s, <4.0s , <5.0s and ≥5.0s for colors of red, green, blue, cyan, magenta and yellow, respectively.

2.6.1 Evaluation

To evaluate the effectiveness of the method, we implemented two naïve rasterization algorithms, and compared to three versions of the our method by varying N, the parameter which determines how many nodes are in each slice (see Table 2.1). The first naïve method, NaïveComposite, computes a batch composite of all the nodes every timestep. The second naïve method, NaïveCoverComposite, rasterizes only the nodes in the cover at each timestep. The primary evaluation metric is efficiency, the time it takes to compute a rasterization. It is also important that the resulting maps correctly represent the structure of the environment. For example, if the map contains a gap in a wall, as a result of incorrectly computing a covering of the environment, robots may be tasked to drive through a wall. In our MAGIC system, the robots have collision avoidance strategies that prevent an accident in such a case. However, attempting to drive through a hallucinated door would slow down the progress of exploration. Therefore, we validated that the maps produced by our approach are structurally accurate (and without gaps), by conducting several real-world exploration missions covering several hours of operation in mixed indoor-outdoor environments.

We evaluated all five algorithm parametrizations on two separate datasets: one involving 10 robots autonomously exploring a section of the Keswick Barracks, and another of 14 robots autonomously exploring a portion of the Adelaide Showgrounds as part of the MAGIC 2010 contest (see Table 2.2 for specific dimensions). While the datasets are both of structured, man-made environments, they have significantly different topologies. This enables us to evaluate performance under varying conditions. The maps corresponding to the final graphs in each dataset are shown in Figure 2.1 and 2.4. Both datasets were taken in Adelaide, South Australia. To our knowledge, no other public datasets of this scale exist.

To evaluate the effectiveness of each method, we ran each algorithm on a sequence of graphs for each dataset, collected at 10 seconds intervals. We evaluated each

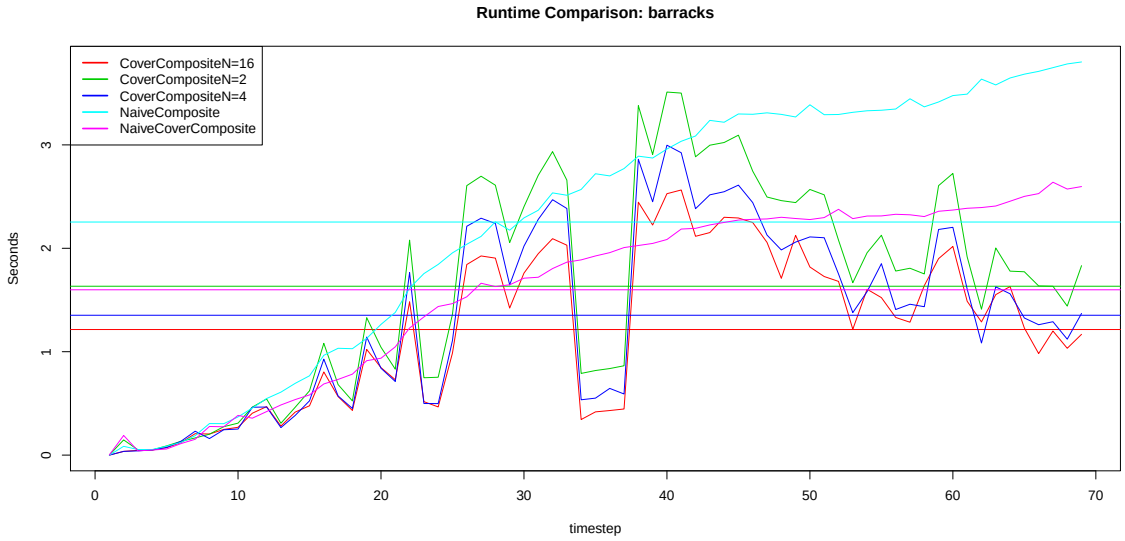


Figure 2.6: Keswick timing detail. Runtimes for each graph $G_0 ... G_{67}$ on the Keswick dataset. Horizontal lines indicate average runtime for that algorithm. Large runtimes indicate loop closures occurred between the $(i-1)^{\text{th}}$ and the i^{th} timestep, which changed the ML posterior for the position of many nodes.

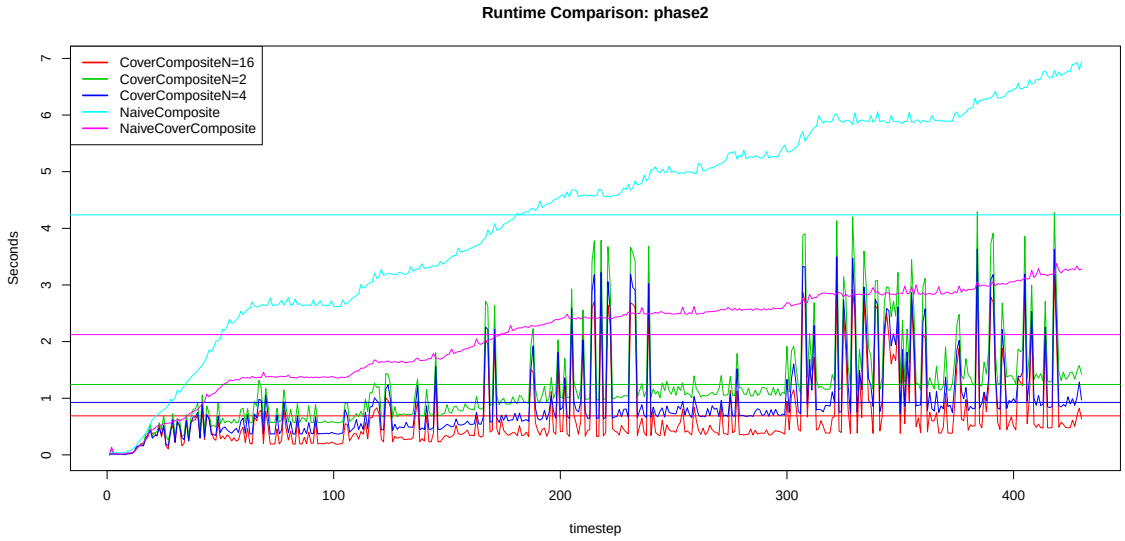


Figure 2.7: Phase 2 timing detail. Runtimes for each graph $G_0 ... G_{429}$ on the phase 2 dataset. Horizontal lines indicate average runtime for that algorithm. The graphs are taken at 10 second increments. Vertical spikes are indicative of significant loop closures. The distribution of loop closures is dictated by the environment, and the paths the robots used to explore it. In this dataset, the system is still functioning well after over an hour in a large environment with many loop closures.

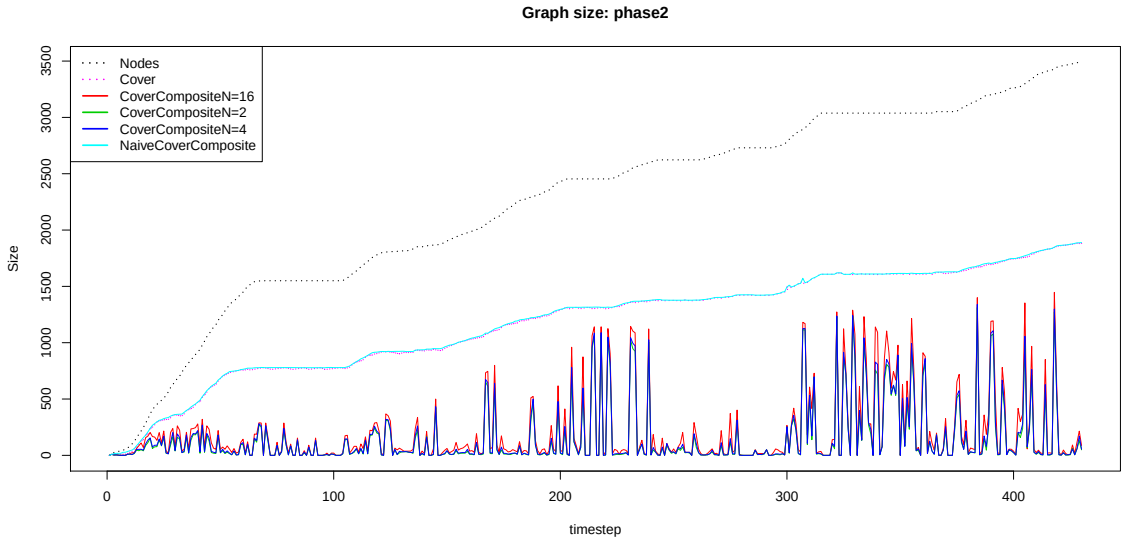
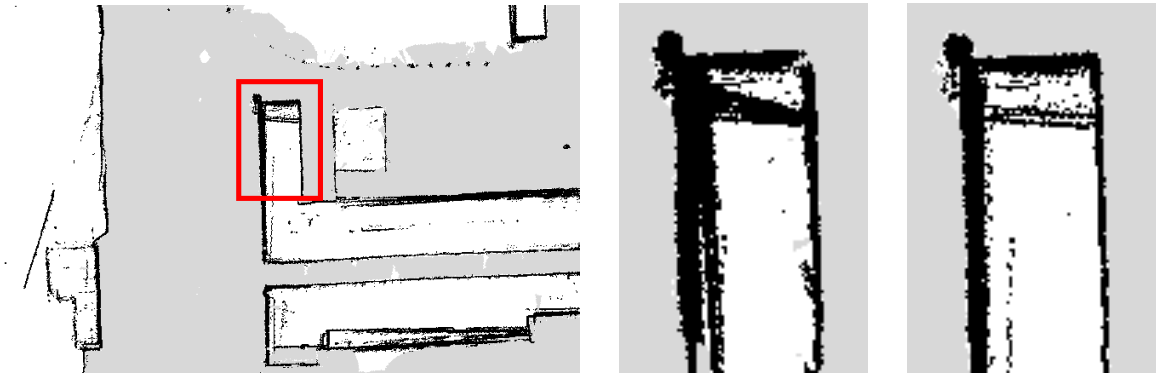


Figure 2.8: Phase 2 node detail. Time varying totals for the number of nodes in the graphs, the number of nodes in the cover, and the number of nodes that must be regenerated for each algorithm at each timestep.

method primarily on the total runtime of the rasterization. All algorithms use the same ML node posteriors, so we expect the maps from each algorithm to be very similar. While it would also be desirable to quantitatively evaluate the map quality from each method, as in [21], we lack accurate ground truth for the environments in question (see Figure 2.4). In practice, we found that our cover heuristic consistently produced structurally-accurate maps over a number deployments in real-world urban environments. Subjectively, we even saw a slight improvement to map quality using the covering method, as can be seen in Figure 2.9.

For evaluation, we timed the duration of each map update for each method on a 2.53 GHz Intel i5 laptop. All our algorithms are hand-optimized, implemented in Java and single threaded. To determine which algorithm is the most CPU efficient, consider the integral time over the phase 2 dataset shown in Figure 2.5 (lower is better). The plot shows that although $N = 16$ has the lowest total time, the $N = 4$ parametrization actually has more instances finishing in under 1.0 seconds (red) than any other method. For MAGIC 2010 exploration missions, despite slightly slower



(a) Context (b) Original (c) w/ Cover

Figure 2.9: Map quality improvement from covering. Using a node covering for a problematic section of the phase2 map reduces feature blurring while still providing a structurally equivalent map. The resulting map more accurately represents the true environment, but without needing to process every collected submap.

average performance, we choose to use $N = 4$, since it reduces the planning delay the most often. Other system designers may prefer to choose a different parametrization, with different runtime characteristics, to maximize the end-to-end performance in their domain. Note that when large scale loop closures occur, we must be willing to incur a heavier planning delay if we want a map which accurately reflects the MLE position of each sensor scan. This is because any cached rasterization is largely useless when the underlying SLAM graph changes significantly. This plot also shows that the CoverComposite algorithm out-performs both naïve implementations.

More detailed timing information for both datasets is available in Figs. 2.6 and 2.7. These plots show the time required by each method to rasterize the graph at each timestep in both datasets. The average time for each method is shown as a horizontal bar. The naïve algorithms show steadily increasing compute time as the number of nodes in the graph (or cover) grows, which is expected. The cached CoverComposite algorithm has variable runtime depending on the number of slices that need to be recomputed. Figure 2.8 shows the number of nodes in the graph, the size of the cover, and the number of nodes that are recomputed at each step. Comparing Figs. 2.8 and 2.7 confirm our claims that the CoverComposite family of algorithms

only incurs significant CPU time when large numbers of nodes have moved significantly due to a loop closure. The data show that our method is significantly more efficient than our highly optimized naïve algorithms we compared against.

2.7 Discussion

Our work in improving the rasterization algorithms using sparse SLAM-graph-based approaches helped to make the state estimation scale to larger environments, which was crucial to the success of our MAGIC entry. This work also demonstrates that occupancy-grid-based representations can be scaled to multiple robots, and that such a representation, which was traditionally used in particle-filter-centric approaches, can be used effectively in combination with sparse, parametric inference methods. However, the presented rasterization approach depends on reliable inter-robot communication to function properly. If local sensor data can't be received in a timely fashion, the concept of a live, globally-coherent map is impossible.

Our MAGIC system relied heavily on a centralized, "reliable" communication infrastructure, which ultimately underperformed our expectations by averaging only 108 bytes per second per robot at a range of about 250 m [2]. This centralized system was also augmented by an opportunistic, best-effort mesh network with substantially higher throughput. This system would likely not scale to larger environments though, as the centralized radio network would have even lower bandwidth at longer ranges, and the opportunistic mesh network would be less likely to maintain all the links necessary to collect sensor data from every robot.

Ad-hoc wireless mesh networks, built from multiple, short, high-bandwidth links are capable of reliable communication, provided the robots can understand the signal strength propagation and then act to prevent the network from becoming disconnected. This limitation motivates the signal strength mapping work in the next chapter.

CHAPTER 3

Signal Strength Prediction for Teams of Robots[1]

Our experience in the MAGIC contest demonstrates that mapping the physical space in which robots are deployed is not enough to allow proper operation of multi-robot teams in large environments. Choosing more powerful radios is not always an option, so it is important that robots can plan to remain connected to support collaboration. One way to do this is to predict signal strength measurements in places to which robots are considering moving. Therefore, we introduced a signal strength modeling and prediction framework in realistic indoor/outdoor urban environments with an *a priori* unknown map [22]. This work extends previous works that typically focus on the case where robots communicate with an existing fixed base station [23]. Unfortunately, many domains lack a usable communications infrastructure (e.g., disaster zones), forcing robots to deploy their own. Achieving good performance from fully-mobile networks is challenging because more complicated signal-propagation models must be incorporated into the path-planning process to ensure connectivity. Furthermore, existing models for fixed-transmitters do not extend directly to the case where all nodes are mobile.

While prediction in the former case is analogous to regression in a two dimensional space, the latter requires making predictions for a four dimensional space, but

[1] ©2012 IEEE. Adapted with permission, from Johannes Strom, Edwin Olson, "Multi-sensor ATTenuation Estimation (MATTE): Signal-strength prediction for teams of robots" September 2012.

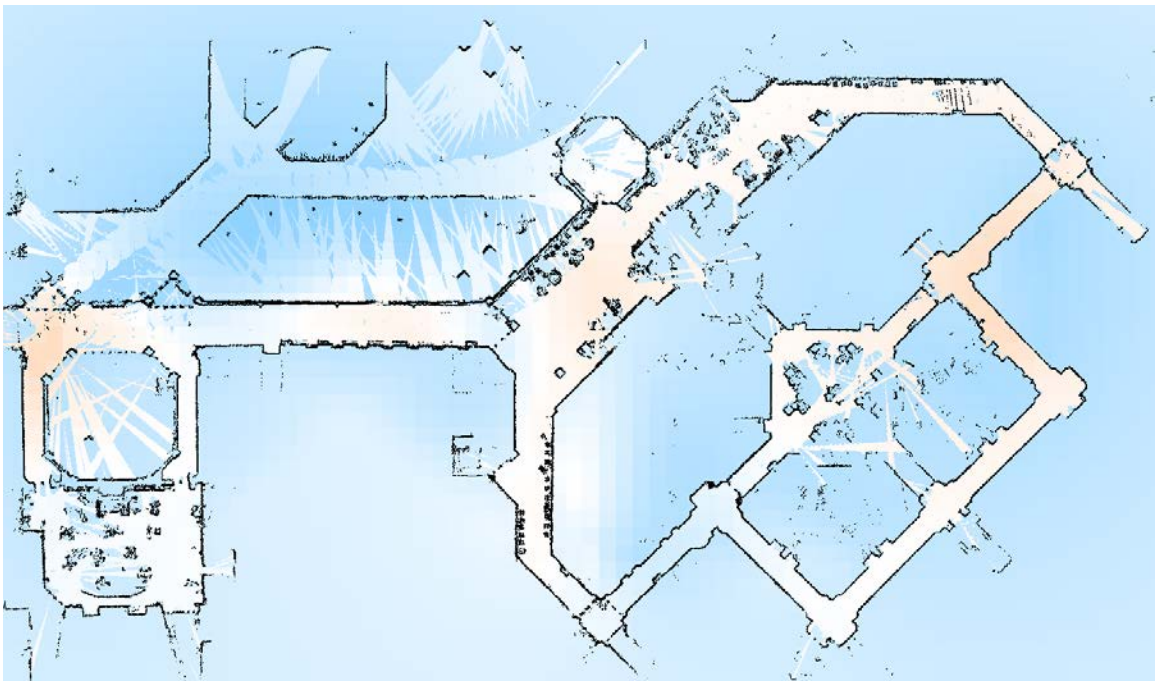


Figure 3.1: Estimated attenuation map. A single traversal of a 160×100m environment by 3 robots conducting an exploration mission allows the environmental attenuation to be estimated. Blue indicates regions where signals are attenuated, white and orange regions indicate where signals pass more easily. Black denotes building structure. Our algorithm, MATTE, is designed for predicting signal strength in the challenging case where all transmitters and receivers are mobile.

without a corresponding increase in data density. The result is that achieving similar generalization performance from observed signal-strength measurements becomes more challenging. This work explores how existing methods can be modified to better cope with this reduction in data density, and introduces Multi-sensor ATTenuation Estimation (MATTE), which uses additional sensor data to inform a better prior over the locations of significant attenuators in the environment. We evaluate the performance of MATTE on over 40,000 m^2 of map and signal strength data collected in a number of environments including academic buildings, libraries, and residential apartment complexes. The scope of our real-world evaluation is several orders of magnitude larger than those presented in existing works, which in one case is 70 m^2 of real world data [24], and less than 1,600 m^2 in a building basement [25].

3.1 Background

Standard macroscopic models of RF propagation describe the expected signal strength, which depends on the location of the receiver $\vec{r}$ and transmitter $\vec{t}$, as having three main components [26]:

$$y_{dBm} = \underbrace{L_0 - a\log_{10}(||\vec{r} - \vec{t}||)}_{\text{path-loss}} - \underbrace{g(\vec{r}, \vec{t})}_{\text{shadowing}} - \underbrace{\epsilon}_{\text{multipath}} \tag{3.1}$$

In simplified, ideal environments, signal-strength can be determined purely by path-loss, which has two degrees of freedom: L_0 corresponds to the power of the transmitter and a is the path-loss exponent that determines how quickly the signal attenuates with distance. In real environments both shadowing and multipath can additionally affect the signal strength: shadowing corresponds to the attentuation a signal experiences as it passes through dense objects, and multipath corresponds to amplification or cancellation that occurs when waves travel multiple paths of different lengths from source to destination. In general, multipath is very difficult to predict because it results from complex reflection and diffraction interactions. Shadowing, on the other hand, is easier to predict, since the scale of its effects are larger and more spatially coherent. For the remainder of this chapter, we will focus on models that predict path-loss and shadowing but ignore multipath. Properly modeling and predicting the effects of multipath is not realistic for multi-robot systems which only sparsely sample the space of possible signal strength estimates.

Robots are able to measure the signal-strength from each other robot during the robot's mission. These noisy values form a vector $\hat{y}$, with corresponding vectors of positions R and T containing all pairs of receiver and transmitter positions, respectively. We treat predicting y as a linear regression problem. For the case of the simple path-loss model, estimating the two variables (L_0, a) from the data is sufficient to predict signal measurements for arbitrary positions. This can be done using standard

least-squares approaches: if $x = [L_0\ a]$ and the i^{th} row of A is $[1\ \ \log_{10}(||\vec{r}_i - \vec{t}_i||)]$ then the least-squared error estimate for x is $\bar{x} = (A^T A)^{-1} A^T \hat{y}$. Path-loss-only models are useful due to their simplicity and correspondence with theoretical propagation equations: the low degree of freedom reduces the chance of over-fitting. However, as this model ignores shadowing effects, its application in environments with varying attenuation is limited. In our evaluation, we will use this simple log-fit as a baseline.

Correlative and tomographic methods include the same path-loss model, but also explicitly incorporate shadowing, allowing for improved predictive performance. The way in which shadowing is captured varies between the two types of models. In the tomographic case, the shadowing function, $g(\cdot)$ is computed by integrating the effect of all attenuators between transmitter and receiver. In practice, we model the world as a 2D grid of infinitely tall columns, represented using the standard tomographic methods as 2D grid of pixels. If the signal passes through a series of p discretized pixels, the shadowing is computed as [24]:

$$g(\vec{r}, \vec{t}) = \sum_{i}^{p} w_i v_i \tag{3.2}$$

where v_i is the attenuation in the i^{th} pixel and w_i is the importance weight of that pixel for that signal's path. (In our implementation, the weights w_i correspond to the length of the line between transmitter and receiver that is contained in the pixel.) In other words, we compute attenuation per pixel by assuming that signal strength is reduced according to the sum each pixel appearing along the path between receiver and transmitter. Given a set of signal strength measurements, we can attempt to find the attenuation values x_i of each pixel using a similar least-squares approach as described above. However, there are many possible attenuation assignments to the pixels that can adequately explain the signal strength measurements, resulting in an under-determined system of equations. In the next section we will discuss apply-

ing regularization techniques to encode a prior that prefers real-world environments, thereby over-constraining the system of equations, and making the estimation of v_i feasible.

In the correlative case, the shadowing component is considered to be uniformly correlated in all directions, allowing predictions to be made by making inferences from spatially-proximate training points. In particular, prior approaches have had success modeling shadowing using a Gaussian-process (GP) [25, 26].

In the case of a fixed base-station (located at $\vec{b}$), standard practice is to use the log-fit as a mean function, and then use a standard squared-exponential kernel to specify the expected covariance between signals at two locations, x and x': $k(x, x') = \sigma_f^2 \exp\{-\frac{||x-x'||}{l^2}\}$. Using this covariance (kernel) function, we can apply standard GP regression techniques to make predictions for a set of sample points [27]:

1. **Compute Log-fit**: Fit L_0 and a using least-squared approach described above.

2. **Fit Hyperparameters**: Choose correlation distance l and function variance σ_f to maximize likelihood of training data.

3. **Compute Covariances**: Compute covariance matrix K_y of training data, and covariance vector k_* of sample points with respect to training data. $K_y = K_f + \sigma_d^2 I$, where each entry $k_{i,j} \in K_y = k(x_i, x_j)$.

4. **Evaluate prediction**: $y_{\text{sample}} = k_*^T K_y^{-1}(y_{\text{observed}} - y_{log})$

These correlative methods have good predictive performance, especially in the case when many training points are available. The main shortcomings of this method are that prediction assumes signals are uniformly correlated in all directions — an assumption that breaks down in the presence of discrete attenuating objects. Direct extension of this method to the case of mobile nodes is also problematic, since training points are in $\mathbb{R}^4$, requiring significantly more data to achieve the same performance.

Finally, this method is computationally expensive, requiring the inversion of a dense matrix whose dimension is determined by the number of training points (typically 600).

3.2 Methods

In this section, we extend both the tomographic and correlative methods to the case of multiple mobile nodes. We will describe our modifications to these existing approaches, and introduce our new approach, MATTE, which uniquely leverages other sensors to infer the location of attenuating objects.

3.2.1 Correlative Methods for Mobile Nodes

Extending previous approaches of correlative prediction to the case of moving nodes exacerbates the data-sparsity problem. Instead of producing signal predictions for $\mathbb{R}^2$ (all points in the plane), we now must produce predictions in $\mathbb{R}^4$ (all possible *pairs* of points in the plane). Since we can't increase the number of signal-strength observations that robots make without slowing down the speed of exploration, this means we have significantly reduced data density. However, we were able to mitigate this problem somewhat by recognizing that our signal-strength models are symmetric with regard to where the transmitter and receiver are — that is, we assume the signal strength is the same from robot A to B as it is from robot B to A. This observation allows us to construct a symmetric distance function, d_{s4}, which effectively doubles the data density:

$$d_{s4}(\vec{r}_a, \vec{t}_a, \vec{r}_b, \vec{t}_b) = \min \begin{cases} ||\vec{r}_a - \vec{r}_b|| + ||\vec{t}_a - \vec{t}_b|| \\ ||\vec{r}_a - \vec{t}_b|| + ||\vec{t}_a - \vec{r}_b|| \end{cases} \tag{3.3}$$

In other words, d_{s4} is a distance metric for pairs of lines that is invariant to rotations of 180 degrees.

In practice, many thousands of observations may be available for use in training the Gaussian process. Due to the $O(n^3)$ computation cost of matrix inversion, it quickly becomes impractical to include all training data. However, prediction performance is improved as more training points are used. In the case where K is very sparse, the inversion can be done more quickly, enabling use of more training data. However, the squared exponential kernel is not sparse; two measurements will never have a covariance of exactly zero, even if they are very far apart. By modifying the kernel to have compact support — that is, the kernel is exactly zero once some distance Θ is reached — the matrix K becomes sparse [28]:

$$k_s(x, x') = \max(0, 1 - \frac{||x - x'||}{\Theta})^\gamma * k(x, x') \tag{3.4}$$

The resulting sparsity allows computing K^{-1} much faster, enabling the use of more training points. Although sparse kernels generally have worse performance, we found in practice the that the speed improvement enabled the incorporation of enough extra training points to achieve a net improvement in performance. This sparsification also increases the number of hyper-parameters that must be estimated to a total of 5. In principle, we can learn these additional parameters the same way we learn the parameters for the original covariance function. However, increasing Θ will always result in a lower training error and an increased computation due to a less-sparse covariance matrix. To limit worst-case computation time, we do an offline parameter sweep to estimate the best Θ and γ subject to a fixed CPU budget.

Together, the symmetric distance metric and the forced sparsification of the correlation function enabled us to adapt existing correlative prediction techniques to the case of multiple mobile robots, and achieve results competitive with our method.

These additions to existing methods form the Multi-robot Gaussian Process "MRGP" method, which we include in our evaluation.

3.2.2 Multi-robot Tomography (MRT)

Conceptually, extension of the tomographic methods to the case of multiple robots is relatively straightforward. The attenuation of each pixel through which a signal passed is estimated in least-squares fashion. The number of pixels, p, is determined by the grid size such that the pixels completely fill the workspace of the robots. For a grid of width w and height h, $p = w \times h$ and pixels are labeled $v_{0,0} \cdots v_{w-1,h-1}$. In addition, the two path-loss parameters, L_0 and a are also estimated simultaneously, resulting in $n = p + 2$ unknowns: $x = [L_0\ a\ v_{0,0} \cdots v_{w-1,h-1}]$. Each of the m signal strength observations y_i provide one equation partially constraining a subset of the pixels in addition to the path loss parameters (see Eqns 3.2 and 3.1). Together these equations are stacked to form the rows of an $n \times m$ matrix A. If A is full rank, $x = (A^T A)^{-1} A^T y$. However, even though m is generally greater than n, A remains rank deficient, resulting in many possible solutions for x. Related approaches have solved this by using Tikhanov regularization that enforces smooth changes in attenuation between neighboring pixels [24]. This corresponds to constructing a Tikhanov matrix Γ whose rows correspond to equations of the form $v_{i,j} - v_{i+1,j} = 0$ and $v_{i,j} - v_{i,j+1} = 0$ for each $i, j < w, h$. The over-constrained solution for x now takes the form

$$x = (A^T A + \lambda^2 \Gamma^T \Gamma)^{-1} A^T y \tag{3.5}$$

where λ is a parameter to determine the weight of the smoothness constraints in Γ relative to the observation equations in A. In contrast to the correlative methods, which scale $O(m^3)$ with respect to the number of observations, the tomographic methods scale $O(p^3)$ with respect to the number of pixels. However, unlike the correlative

methods, $A^T A$ is naturally sparse since each pixel is only jointly constrained with a small number of other pixels. This means that sparse matrix-inversion methods, such as a Cholesky decomposition, can perform significantly better than $O(p^3)$. In practice, $A^T A$ is still dense enough (95% zeros) that we are limited to solving grids on the order of 100×100 (10000 pixels). For the datasets in our evaluation, this translates to a grid size between 2 and 4 meters. Such large grid sizes poorly approximate the sharp spatial changes in attentuation found in real environments, such as at the border between a building and neighboring free space. Furthermore, in order to avoid over-fitting, λ must be set large enough to allow very little spatial gradient in the attenuation of each pixel (see Figure 3.1). In areas where attenuation changes quickly, the predictive power is limited. Our implementation of this approach, Mutli-robot Tomography (MRT), is included in our evaluation.

3.2.3 Multi-sensor ATTenuation Estimation (MATTE)

The approaches we've discussed so far focus solely on using signal-strength readings for prediction of future measurements. Many robots already carry additional sensors for mapping and navigation. Intuitively, since the physical structure of an environment (the position of walls and other solid surfaces) influences signal propagation, a map of the environment should help predict signal propagation. This is especially true for sensors like laser range-finders, which tend to have a range of at least 10–30 meters. Incorporating a map does not solve the problem completely though, since the attenuation of a wall depends on the material and LIDAR generally can't distinguish between a reinforced concrete wall and drywall.

Specifically, we propose the use of occupancy grids derived from laser range-finders to provide a more informed regularization constraint to tomographic methods. The occupancy grids collected by our robots label the world with three classes: known free space, known structure and unknown. This information can provide a much

better prior about the attenuating properties of the environment. For example, we generally expect areas that are marked as free space to pass signals with little interference. Similarly, knowing the location of structures can provide a prior about where attenuation should increase dramatically. Besides providing a better prior about the magnitude of attenuation, the occupancy grids also provide a much finer view of the environment. Using the MRT method, we are typically limited to coarse grid sizes (e.g., 3 m for our datasets) due to computational constraints. On the other hand, LIDAR-based occupancy grids can be computed cheaply even for fine-grained grid sizes (we used 10 cm grid in our experiments).

Our approach explicitly estimates the attenuation of each of the three classes separately. A simple approach to this problem is to fit a single attenuation value to all pixels of the same class. For example, known free space might have an attenuation of $-0.01dBm$ per meter, whereas structure (e.g., walls) could have an attenuation of $-0.3dBm$ per meter, and unknown space could be approximated as somewhere in between. This approach is notable in its simplicity — it has no parameters to tune and only 5 degrees of freedom to fit the observed data: two for path loss parameters and three more for the attenuation assigned to each class in the occupancy grid. In general, however, it is a poor assumption that all objects detected by the robots as structure will have the same attenuation. For example, a wooden fence and a brick building have very different effects on a signal, but can appear very similar to a robot's laser range finder. As our initial experiments confirmed, this model performed poorly in explaining real-world data.

A better way to use the occupancy grids is to better inform regularization methods, for example by enforcing smoothness constraints only between neighboring pixels of the same class. Using a more flexible model is difficult, however, since individually estimating the attenuation for each pixel in the fine-grained occupancy grid results in a poorly constrained, computationally prohibitive problem. The largest map we

present has nearly 5 million pixels. Downsampling to a coarser resolution can make the problem more tractable, but reduces the ability to incorporate features such as doors or walls, which may be lost by resampling.

Algorithm 3.1 MATTE_UPDATE(trainPoints, map)

init A, b from $bounds(map)$
for all $p \in$ trainPoints **do**
  $r \leftarrow [1, \log_{10}(||p_r - p_t||), 0 \cdots, 0]$
  for all locations $v_i \in$ losPath(p) **do**
    $c \leftarrow map.class(v_i)$
    $r(v_i, c) \leftarrow 1$
  end for
  $A \leftarrow appendRow(r)$
  $b \leftarrow appendElement(p_z)$
end for
$\{A, b\} \leftarrow appendRegularization(A, b)$
$x \leftarrow cholesky.solve(A^T A, Atb)$
store x
store map
return

Algorithm 3.2 MATTE_PREDICT(testPoints)

retrieve x
$z \leftarrow$ vector$(testPoints.len)$
for all $p \in$ testPoints **do**
  $z_p \leftarrow x(0) + log_{10}(||p||)$
  $z_s \leftarrow attenuationIntegral(x, p, map)$
  $p \leftarrow$ append$(p, z_p + z_s)$
end for
return p

Instead, we introduce a method that benefits from the detailed resolution of the map, but has computational complexity comparable to the MRT method. Our technique, called MATTE, separately estimates spatially-varying attenuation for each of the three classes in the occupancy grid. These estimates allow querying any point in the environment and determining, for example, *if* there was structure there, what attenuation it would have. When predicting signal-strength, we first examine pixels in the occupancy grid to determine which class they are, and then perform a look-up

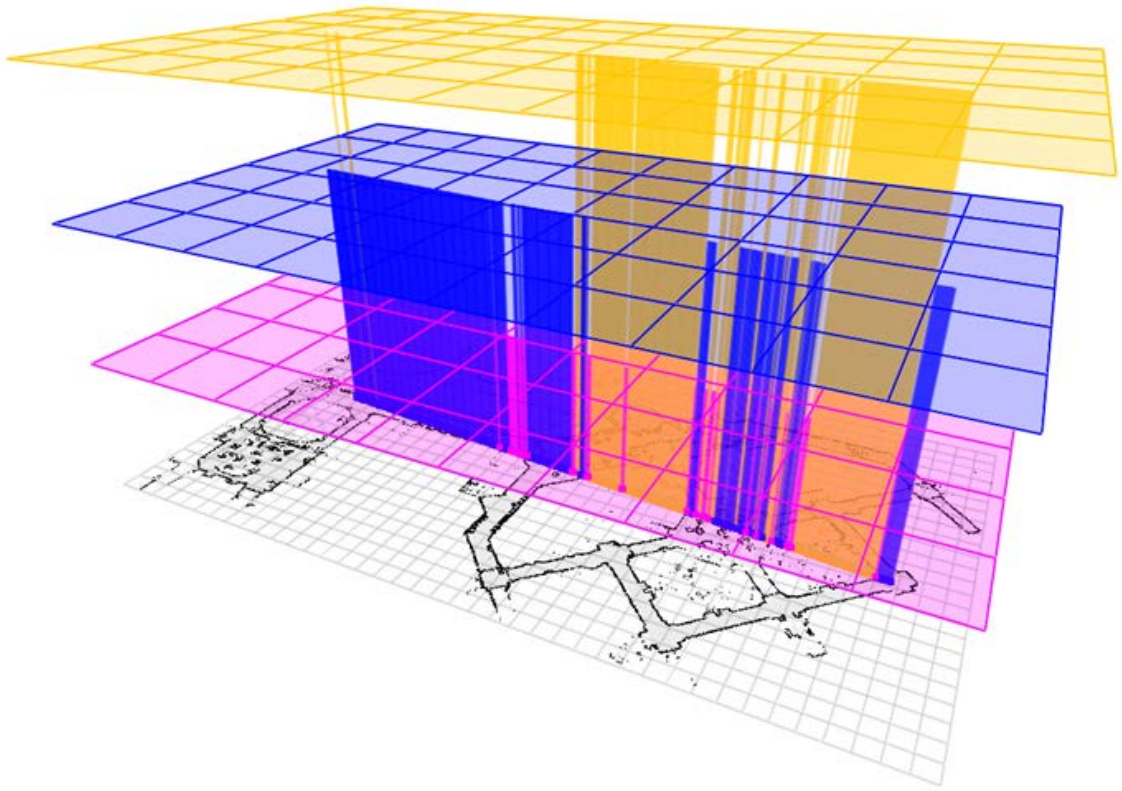
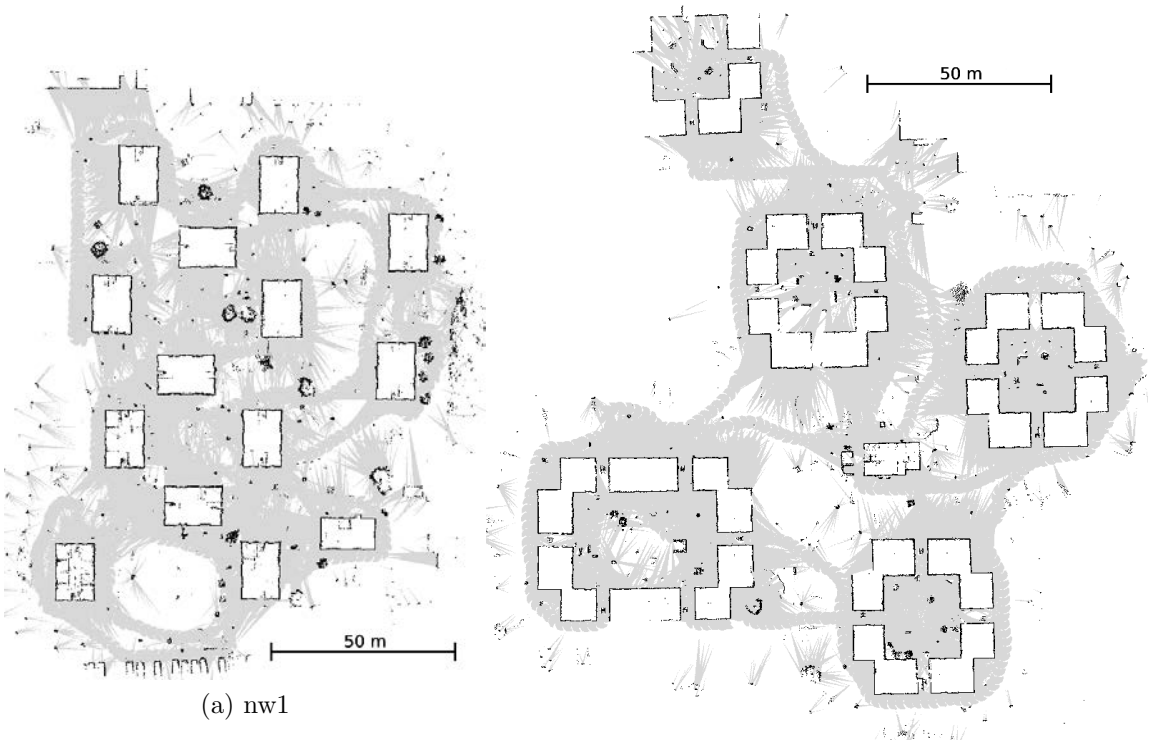


Figure 3.2: Attenuation integral using the three-class representation. An estimate for $g()$ is obtained by integrating over the cell-by-cell attenuation estimates from the receiver to transmitter. Using three coarser grids, one each for free, unknown, and occupied space, was crucial to efficiently estimating the attenuation while still leveraging fine-grained representation of the environment. The illustrated line integral passes through known free-space, known occupied space, and unknown space, denoted by blue, pink, and yellow, respectively. For each cell in the high-resolution occupancy grid, the corresponding attenuation is incorporated via a lookup into a lower-resolution grid of class-specific attenuation estimates.



(a) nw1

(b) nw2

(c) bbb3

(d) eecs4

(e) dc1

Figure 3.3: Five evaluation environments for MATTE. From left to right they are: two outdoor sections of the Northwood residential housing complex, the 3rd floor of the Bob and Betty Beyster Building, the 4th floor of the Electrical Engineering and Computer Science Building and the first floor of the Duderstadt Library. White is unexplored, gray is known free-space and black is structure. Each evaluation set consists of back-to-back traversals of three of these environments; meta-parameters are tuned using the remaining two. These maps are rasterized using the techniques presented in Chapter 2.

on the appropriate attenuation estimate, as shown in Figure 3.2. The benefit of this approach is that we can estimate the spatially-varying attenuation at a resolution independent of the resolution required to adequately map the environment structure. Typically grid sizes of 10 to 20 cm are required to preserve the presence of doors or thin walls — but attenuation rates of a particular material, (e.g., building) tend to vary at a much slower rate. The computational implications of this approach are significant — whereas the resolution of the occupancy grid is typically 10 cm, we have achieved good results with a grid size of 4 m for the spatially varying per-class attenuations, resulting in a reduction by a factor of 1600 in the number of unknowns.

Similar to the previous tomographic approach, we simultaneously estimate the path-loss parameters and the spatially varying attenuation. For a workspace $w \times h$ pixels in dimensions, we estimate a total of $2 + 3 * w * h$ variables that comprise x. Although this represents a threefold increase in the number of degrees of freedom over our previous approach, $A^T A$ is generally more sparse (e.g., 99% sparse in our datasets), ultimately requiring less time to compute a result. An overview of the update and prediction steps are shown in Algorithms 3.1 and 3.2. The MATTE algorithm has several important parameters which can either be tuned by hand or estimated automatically from training data. The most important parameter is the relative weight λ given to the smoothing regularizer we described in Eqn. 3.5. In addition we introduce three parameters to govern the expected attenuation in areas where we have not collected any signal data: a weighting factor ϕ determines the relative weight of this prior, and prior attentuation values in units of dBm per meter for each class are ρ_f, ρ_s, ρ_u, for free-space, structure, and unknown space, respectively. In practice, we set $\rho_s = \rho_u$, allowing free space to have a distinct prior from other areas.

An additional potential advantage of MATTE is that new occupancy grids can be incorporated cheaply, as the underlying spatially varying attenuation does not neces-

sarily need to be updated when the map changes. For example, our approach might encode the knowledge that a pixels labeled as "structure" in a particular area tend to have an attenuation of X dBm per meter. If the map of that area is later expanded, that information will be able to provide an estimate for the attenuation expected there, without needing to recompute x. This allows our approach to potentially be adapted to run online.

Evaluation	Method	Meta-parameters
1	Training: {nw2,bbb3}	Testing: {eecs4,dc1,nw1}
	MATTE	λ=2.0 ϕ=.2 ρ_f=-.1 $\rho_{u,s}$=-.3
	MRT	λ=0.1375
2	Training: {eecs4,dc1}	Testing: {bbb3,nw2,nw2}
	MATTE	λ=1.225 ϕ=0.05 ρ_f=0.2125 $\rho_{u,s}$=-0.2875
	MRT	λ=0.1125
3	Training: {bbb3,eecs4}	Testing: {nw2,nw1,dc1}
	MATTE	λ=2.0 ϕ=0.275 ρ_f=0.2875 $\rho_{u,s}$=-0.4875
	MRT	λ=0.1250
4	Training: {nw1,eecs4}	Testing: {nw2,dc1,bb3}
	MATTE	λ=1.38 ϕ=0.3 ρ_f=-.1 $\rho_{u,s}$=-.3
	MRT	λ=0.1125
5	Training: {dc1,nw1}	Testing: {eecs4,bbb3,nw2}
	MATTE	λ=2.0 ϕ=0.1 ρ_f=-0.15 $\rho_{u,s}$=-.3
	MRT	λ=0.5875

Table 3.1: Signal-strength prediction parameters. Parameters are automatically determined for each evaluation using coordinate descent on the training set. Some parameters were fixed for performance reasons, including the grid sizes for MATTE and MRT methods, at 4.0 meters and 3.0 meters, respectively. For MRGP, the number of training points was set to $\min(600, .05 * m)$, Θ was fixed to 20.0, and γ to 3. Also σ_d and σ_f were fixed for numerical stability reasons at 1.5 and 1.0, respectively. For MRGP, coordinate descent fixed l to 28.75 on all training data.

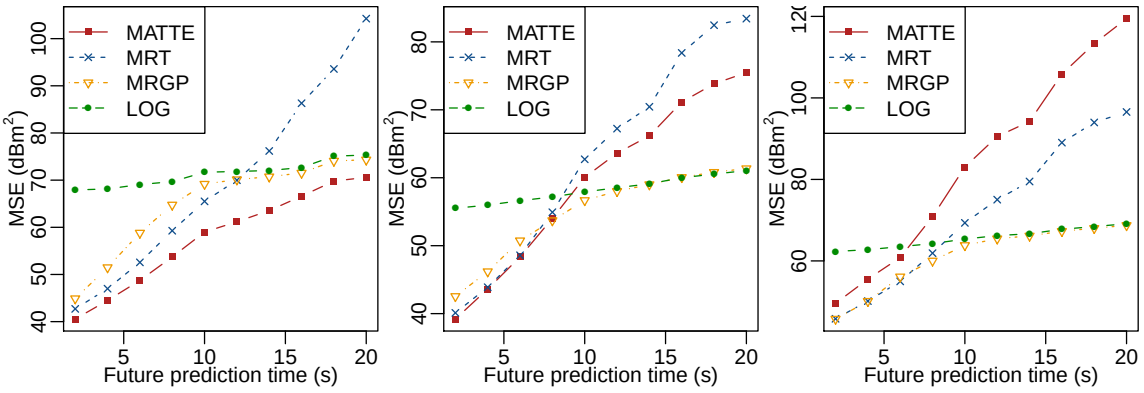
3.3 Evaluation

We evaluated the three primary methods we have discussed to determine which methods were most successful at predicting real-world signal-strength measurements. The main purpose of our evaluation is to show that previous signal-strength prediction methods using a fixed base station can be extended to the more general case where all nodes are mobile. We also show that in many cases sensor data can be effectively leveraged to further improve signal-strength predictions. Several of our models have many degrees of freedom, so our evaluation also seeks to show that the methods we present can generalize well from previous observations to the prediction of future signal measurements.

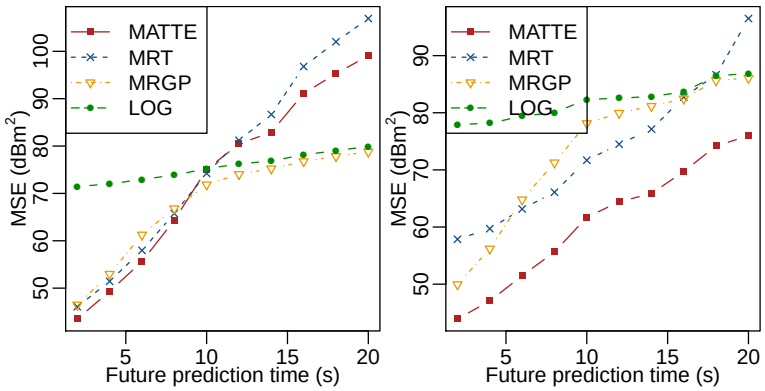
3.3.1 Test Apparatus

We used a robot platform our lab custom designed for urban reconnaissance [2]. We outfitted three of our 14 robots with additional 2.4 GHz TP-Link WiFi radios which were programmed to report signal-strength measurements at 20 Hz. Our robots are equipped with 3D laser range finders, in addition to IMUs and odometers, enabling them to produce high-quality globally consistent maps using SLAM algorithms [18, 29, 30]. This capability allows us to quickly collect large amounts of signal-strength data that is co-registered with a global grid.

We used these three robots to collect a series of 5 datasets in various indoor and outdoor environments, spanning a total area over 40,000 m^2. The datasets consist of three indoor environments, **bbb3**, **eecs4** and **dc1** for short, as well as two larger outdoor environments, abbreviated **nw1** and **nw2** (see Figure 3.3 for details). Real-world rescue robots must operate in mixed indoor-outdoor urban environments. In order to mimic these conditions, we randomly selected a sequence of three of the environments for testing, guaranteeing a mix of indoor and outdoor datasets. Each of the datasets consist of exploration missions so very few sensor measurements can be



(a) Eval 1 {eecs4,dc1,nw1} (b) Eval 2 {bbb3,nw2,nw1} (c) Eval 3 {nw2,nw1,dc1}



(d) Eval 4 {nw2,dc1,bbb3} (e) Eval 5 {eecs4,bbb3,nw2}

Figure 3.4: Evaluation of signal-strength prediction accuracy. MSE is shown for each method on of the five the testing datasets. Error bars are bounded by 0.8 dBm^2 and are omitted for clarity. Our proposed method, MATTE, is compared to extensions of previous methods MRGP and MRT. Our method generally performs better for near-term predictions where the robots' sensor data provides an informative prior. Poor performance of MATTE in evaluation 3 is attributable to the randomly generated training set which contains only indoor datasets, while the testing set contains both indoor and outdoor sets.

considered duplicates, since robots do not tend to retrace their steps. This enables us to explicitly test the predictive performance of each method, rather than their recall abilities.

For each randomly-selected set of three areas, we replayed the signal-strength observations and corresponding maps in two second increments. After each step, the methods were tested on their prediction performance of the next 10 intervals of future signal strength data (from 0–2 seconds, 2–4 seconds, 4–6 seconds, up to 18-20 seconds). As the traversal played out, methods had access to an increasing amount of training data, but the amount of testing data was relatively similar at each step. Several of the methods also have meta- or hyper- parameters, which need to be tuned before the start of a traversal. We used the two remaining areas not selected for the test set to train meta parameters using a compass search. We include results for 5 such randomly generated traversals. The parameters selected by the compass search for each method on each evaluation are shown in Table 3.1. All of the methods we presented were tuned to use comparable amounts of computational resources.

While automatically tuning the meta parameters could be used to tune for a wide variety of use cases (even outside multi-robot exploration), we particularly optimized the parameters to target the online, multi-robot exploration case. While each method exhibits significantly different asymptotic growth, and as such cannot be tuned to have identical CPU usage, we set parameters that resulted in roughly equal CPU use over all datasets. Specifically, cumulative run-times for MATTE ranged between 2 seconds to process the **bbb3** dataset up to 30 seconds for the significantly larger **nw2** dataset. For the GP method, times varied between 15 seconds for **bbb3** datasets and 20 seconds on **nw2**.

3.3.2 Results

The testing results for the five evaluation sets are displayed in Figure 3.4. As expected, all methods are better at making short-term rather than long-term predictions. This is a result of the fact that measurements nearer in the future are more likely to be similar to existing measurements, or the fact that attenuating objects impacting observations in the near future are likely to be correlated with sensor data the robots have just now collected. The evaluation shows that when predicting more than about 8-10 seconds in the future, MRT, MRGP and MATTE perform worse than the simple log baseline. This suggests the approaches presented here are best suited for near-term predictions. For use in a multi-robot exploration system, the performance around 0-6 seconds in the future is of most interest, since the system will continuously be replanning.

All of the methods we have implemented show competitive performance, especially for predictions between 0 and 10 seconds in the future. However, MATTE significantly out-performs the other methods in evaluations 1 and 5. In evaluations 2 and 4, it performs comparable to the tomographic method. However, in evaluation 3, our method exhibits worse performance than the other methods. An examination of the meta-parameters determined via compass search show evidence of over-fitting the training set, which by nature of our randomly selected test sets, happened to both be exclusively indoors. This manifests as a positive attenuation prior for free-space (ρ_f), consistent with the wave-guide effect sometimes seen in hallways. In the other evaluation sets, there was at least one outdoor dataset used for training, which helped to mitigate this type of over fitting. In most of the evaluation sets, we see that MATTE can outperform the correlative method for short-term predictions up to ten seconds in the future. For longer-term predictions, where training data is mostly useless, it is difficult to beat the log baseline. Our evaluation shows that even without explicitly capturing multipath in our models, we can still make meaningful predictions

for robot teams. For example, near term predictions for MATTE have a typical MSE of 40 dBm^2, or RMSE of 6.3 dBm. For signal strength measurements which typically range from 40 to 110 dBm, MATTE can provide a signal strength estimate that can differentiate between good (low) and bad (high) communication links.

3.4 Discussion

Our signal strength framework for multi-robot teams is an important step towards operating in environments with constrained communication, such as the indoor/outdoor urban environments in the MAGIC contest. The signal strength estimates could be, for example, combined with a communication-aware planning framework to ensure the maintenance of a single, connected ad-hoc mesh network for the entire team of robots [31]. However, any planning approach that makes use of the prediction framework described here must be able to ensure the communications graph remains connected. While short interruptions could be tolerated, extended outages would reduce the map and signal strength information available, interrupting the crucial inputs needed to make good predictions. Other system integration challenges also exist; the work here uses two distinct physical radios per robot, one for TX and one for RX. This was primarily due to limitations of Linux "monitor" mode. Better integration with radio firmware could reduce the size of the required hardware, and potentially also improve signal strength estimates.

Mapping the physical and signal-propagation properties of an environment is just one of the many mapping and perception tasks that multi-robot teams must tackle. In the next sections, we discuss how similar multi-variate-gaussian and grid-based tools can be applied to the problems of point cloud segmentation and camera calibration.

CHAPTER 4

Colored Point Cloud Segmentation

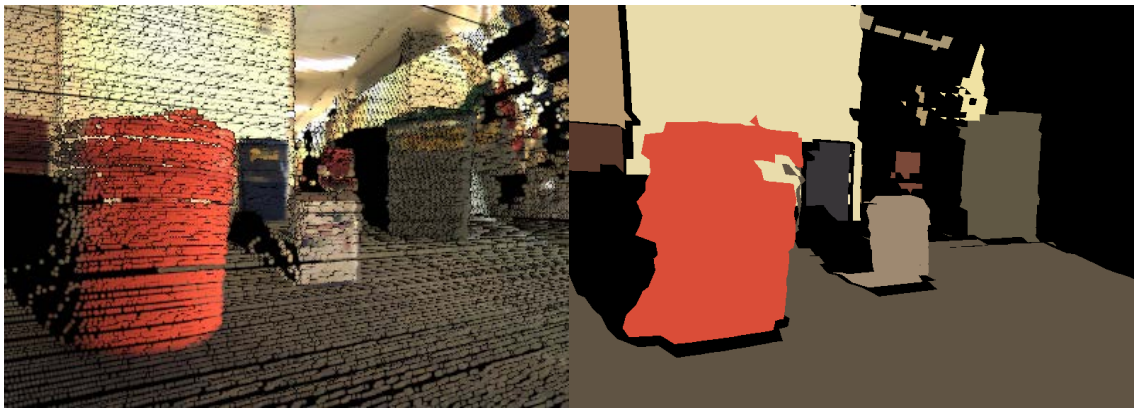
Joint work with Andrew Richardson[1][2]

In addition to mapping the physical layout and signal-attenuation properties of an environment, multi-robot teams are often required to identify particular items of interest, for example IEDs or human survivors [32]. However, detecting such objects (or even much simpler ones such as the colored trash cans in the MAGIC competition) remains a challenge for robots deployed in real, complex, cluttered, indoor-outdoor environments. Similar to the previously described LIDAR-aided signal-strength prediction approach, one strategy for improving detection of objects is to combine data from multiple sensors, such as LIDAR and cameras. This allows analysis of both the shape and visual appearance of the object which can yield better classifications.

However, combining data from cameras and LIDAR introduces several additional challenges, including higher data rate, and data streams of incomparable units (distance vs color). Model-based object detection or complex surface extraction struggles to run online, especially when faced with 3D point clouds. Furthermore, typical depth estimation techniques from the vision community are either high quality and not real-time [33, 34], or real-time at the expense of quality [35]. Laser scanners, however, support acquisition of highly accurate and reliable spatial data without expending

[1] Both Andrew and I contributed equally to this project

[2]



(a) Colored lidar scan (b) True-color segmentation results

Figure 4.1: Sample segmentation. Our method uses leverages both color and surface normals to enable online segmentation.

significant additional computation resources.

A common technique to make processing steams of sensory information tractable in real-time is to segment the data into superpixels [36] or spatially adjacent regions [37]. This data reduction technique enables tractability for terrain classifiers [38], or object classification algorithms [39]. While jointly calibrated laser and camera systems have been used widely for applications such as pedestrian tracking [40] and road following [38], such approaches have all separately considered the visual and spatial components. These approaches segment separately, and then recombine the results near the end of the processing pipeline. Some recent work has also considered the case of colored point clouds in aerial surveying applications [41]. While the work shows the benefits of analyzing the joint data stream, it does not address the online requirement for use on a robot system.

Prior work on segmenting laser point clouds relies either on determining segment boundaries using Euclidean separation or change in local surface normals. The former approach, [37], fails to produce meaningful segments when objects can be connected by a continuous string of points through some common supportive surface (e.g. the ground plane, or a wall). This can lead to improper agglomeration with the support-

ing surface and make later detection or classification difficult. More sophisticated methods use a local estimation of the surface normal for every point as a segmentation criterion [42]. However, they use a $kd-$tree to lookup nearby points, which requires $O(n \log n)$ time to construct. Furthermore, solely determining reasonable segment boundaries based on surface normals can result in over segmentation. This occurs because local surface normal estimation is very sensitive to the range noise when scan density is high. Others have shown success in indoor or simulated environments using a high-frequency, high-precision, low-range, low-field-of-view laser scanner [43]. However, with lower resolution, wider field of view scanners, such as those often used for robot navigation (and in this work), normal estimation becomes more noisy.

Using our co-registered sensors, we demonstrate a robust method that exploits the structure of the laser data in combination with the additional information provided by the camera to speedup and improve the segmentation process. The key contributions in this work are:

- An extension of graph-theoretic segmentation to propose segment unions based on spatial proximity
- A dynamic segment union criterion based on color and surface normals that produces a quality segmentation
- An efficient segmentation process, bounded by $O(n\alpha(n))$, which is a linear bound for all practical n

Quantitative evaluation of segmentation quality is difficult because the effectiveness of the segmentation is determined by the application for which it is intended. In this work, we evaluate segmentation quality based on its ability to distinguish between objects and the surrounding environment.

4.1 Point Cloud Co-registration

Given a set of planar laser scans L, and a corresponding image I, we are interested in computing the joint point cloud of vectors $\vec{v} = [x\ y\ z\ r\ g\ b]^T$ which will form the basis of our segmentation algorithm described in the following sections. Each laser scan L contains a set of range-bearing measurements $R = \{(r_1, \theta_1), \cdots, (r_i, \theta_i)\}$, and an associated rigid body transformations $T_i = T_i(\phi)$ mapping each homogeneous vector $\vec{r_i} = [r_i \cos(\theta_i)\ r_i \sin(\theta_i)\ 0\ 1]^T$ to a point $\vec{l_i} = [x\ y\ z\ 1]^T$ in the sensor's coordinate frame. These rigid body transformations are determined dynamically from the angle of the servo controlling the laser scanner, ϕ. The full experimental sensor configuration can be seen in Figure 4.2.

Producing the joint cloud now only requires projection of the laser points l_i into the most recent image I, and discovering the pixel $\vec{p_i} = [p_x\ p_y]$ with coloring r, g, b that corresponds to l_i, finally producing v_i. Determining the correct pixel $\vec{p_i}$ also depends on the distortion model for the camera, as covered in [44] and [45]. The projection is described precisely by Equation 4.1.

$$p_i = d(KET_i\vec{r_i}) \tag{4.1}$$

where K is the standard 3×3 intrinsics matrix with 5 degrees of freedom (DOF), $E = [R\ t]$ is the 3×4 extrinsics matrix with 6 DOF, T_i is the 4×4 rigid body transformation described above, and $d(\cdot)$ is a sixth degree polynomial approximation of the lens distortion using only 5 DOF.

The production of the joint cloud is highly sensitive to the accuracy of the co-registration between the two sensors described by the 16 parameters in Equation 4.1. Prior methods on co-registering laser and cameras are covered in [46], [47] and [48]. In the first case, calibration is achieved by manually placing specialized calibration targets in the scene. The second case relies on measurements from an IMU directly

Figure 4.2: Sensor configuration. Co-location of the LIDAR and camera enables co-registration of both data streams.

attached to the sensor rig to aid calibration. In the final case, some pre-processing of the laser and camera data is done to aid a human in identifying correspondences manually. For the purposes of this work, we follow an approach very similar to [48]. Our calibration is optimized on a scene-by-scene basis. For each scene, we manually select a set of corresponding points in both camera and LIDAR space. For example, object corners are generally easy to identify in both the camera image and the point cloud. We then run an iterative compass search, varying $k_1 \cdots k_5, e_1 \cdots e_6, d_1 \cdots d_5$, to minimize the mean square projection error:

$$e = \sum_j |p_j - d(KET_j\vec{r}_j)|^2 \tag{4.2}$$

As the calibration is an offline process and our method requires a very accurate calibration, the optimization is allowed to run until an error-minimum is reached.

After calibration, points which project into the image are assigned the appropriate color, as shown in Figure 4.3a. An alternative approach would be to label all the pixels in the image with an x, y, z point. However, the angular resolution of the camera is finer than that of the laser scanner, so we can avoid dealing with a missing data problem by considering the color points in 3D space.

4.2 Graph-based Segmentation

In this section we discuss the extension of a popular graph-theoretic segmentation method for images to the domain of co-registered laser and color points [36]. This method has been extended successfully to many domains, including for use in parallel segmentation [49] and to tracking fiducial tags [50]. Given the joint cloud $V = [v_1, \cdots, v_i]$ whose construction is specified above, we consider the problem of segmenting (or clustering) this set into disjoint subsets S_i s.t. $\bigcup_i S_i = V$.

We consider a graph (or mesh) $G = (V, E)$ whose vertices V are points from the joint cloud (above), and some set of edges E. In the vision literature, where the nodes of the graph are typically the pixels in the image, E is constructed such that each pixel is 8-connected to all of its neighbors. In recent work on clustering laser clouds, E is determined by connecting all points to all the neighbors within a radius r [37].

Given this graph, there are many ways of clustering nodes together. In [37], the resulting clusters are all the connected components, determined by greedily merging all nodes that are connected by an edge. This approach, variants of which are used also in [42], only incorporates a binary edge criterion. That is, each edge is either present or absent. In [42], edge existence is further screened by a fixed threshold for angular difference in surface normal. Other segmentation criteria, for example making assumptions designed to remove the ground plane, are discussed in [51].

In our method, we also use a real-valued edge weight, but only leave edges up to a specified length intact, generally 0.25 m. This avoids the requirement to special case the removal of the ground plane, and allows segments to be formed that are uniformly variable, while with the same parameters also separating smooth areas from segments that have a high variability. To achieve this, we use as a foundation a segmentation algorithm from [36].

When operating on an image, this segmentation algorithm generates an 8-connected graph, as described above. Edge weights are determined as a function of color error

between neighboring pixels and then edges are sorted in increasing order.

Segments are formed by processing each edge $e = \langle p_i, p_j, w \rangle$ in order, starting with the smallest w. For each pixel p_i and p_j, the corresponding segments containing those pixels, S_l and S_m, are retrieved, in addition to corresponding threshold values $thresh_l$ and $thresh_m$. A merge between S_l, and S_m to form S_{lm} is accepted if and only if $w < thresh_h$ for both $h = l$ and $h = m$. If the merge is accepted, w is guaranteed to be the largest weight between any two pixels in S_{lm}, so we can update the threshold for the newly formed segment as:

$$thresh_{lm} = w + \frac{k_{rgb}}{|S_{lm}|} \tag{4.3}$$

The net effect is that, as the segments grow, the threshold approaches the largest intra-segment weight. The resulting segmentation process ensures that areas of uniform edge cost are joined first, while segments with larger variability are formed after small-weight edges are considered. This enables the highly desirable property of segmenting regions of uniform color, gradients, and, crucially, regions of uniform variability. These results are discussed in more detail in [36].

We extend this method in two ways. First, we replace the image-based grid with a surface mesh constructed directly from successive laser scans. Each point in the joint cloud is connected by a potential edge to its 8 neighbors in the current, previous and subsequent scans. While the segmentation depends heavily on the actual edge weights, a significant segmentation improvement can be obtained by simply building a mesh that does not contain an edge from an object in the foreground to one far away in the background. This extension better enables separation of similarly color objects from a background, which the image based grid often cannot handle.

Second, we consider the case where there are two weights for every edge: one for color difference and one for angle difference between surface normals. Integrating

a second edge weight into the scheme described above raises two important issues. Both issues stem from the fact that weights for the surface normals and weights for color are in fundamentally different units and cannot be directly compared. The first problem is that sorting simultaneously on two distance metrics is ill-defined. One could conceivably define a mixture of the two measures and sort on the combined weight. However, this introduces an additional parameter that must be tuned, and requires finding appropriate normalization schemes for both the color distance and the surface normal distance metrics.

When considering how to sort the edges, runtime is an important concern. If the edges are real-valued, the best sorting performance is $O(n \log n)$. However, in the case of the discrete labels (e.g. in RGB), the number of possible distances is constant (though potentially large). This allows a constant time sorting algorithm to sort edge weights in $O(n+k)$, where k is number of possible distances. Since n is very large for a laser point cloud, sorting real-valued edges quickly becomes prohibitive. Because sorting on both edge weights is ill-defined, we choose only one of the edge weights to sort on. Since the edge weights corresponding to color are more stable than local surface normal estimation, and since they are also discrete, we can continue to use counting sort in $O(n + k)$ time.

Note that sorting edges without incorporating surface normal weights does not provide the same theoretical guarantees shown in the color-only case [36]. However, our algorithm still yields a significantly improved segmentation in the environments presented in the evaluation. As noted above, sorting on both criteria is ill-defined.

To integrate the edge weight corresponding to the surface normals, we add an additional threshold in similar fashion to the camera-only approach described above, k_{norm}. This means that for an edge $e = \langle l_i, l_j, w_{rgb}, w_{norm} \rangle$ to connect two segments S_l, S_m into a larger segment S_{lm}, we need the following to hold: $w_{rgb} < thresh_{rgb}$ and $w_{norm} < thresh_{norm}$ for the thresholds corresponding to both S_l and S_m.

Similar to [36], we employ an 8-connected graph for segmentation. This graph is extracted from the mesh of laser scans. Only laser points that have a corresponding color value are included in the mesh. Each node in the surface mesh is connected to up to 8 others by edges labeled with two edge weights. The first edge weight, discussed above, is simply the Euclidean distance in the RGB color space. The RGB color space is chosen over HSV or other alternative color spaces because it corresponds to the native color space used by our camera, and because it was used effectively in prior work [36]. The color edge weight between two pixels $\vec{u}, \vec{v}$ is then given as:

$$w_{rgb}(\vec{u}, \vec{v}) = \sqrt{(u_r - v_r)^2 + (u_g - v_g)^2 + (u_g - v_g)^2} \tag{4.4}$$

Computing the corresponding edge weights for surface normals is slightly more involved because it involves computing the surface normal at each point. In practice, this can be done with reasonable accuracy by only looking at the neighboring points. A comprehensive study of local normal estimation techniques is presented in [37]. For this work, we find the best fit plane for the 8 neighbors using a Singular Value Decomposition (SVD) because it was shown to have the highest accuracy and a reasonably fast runtime [37][3].

Let $\vec{q}_1 \cdots \vec{q}_m$ be the m points connected to point $\vec{q}_0$ by an edge, where m is necessarily no greater than 8. Defining the mean of the points to be $\bar{q} = \sum_j \vec{q}_j / m$, then we can pack a $m \times 3$ matrix Q, such that row j corresponds to $\vec{q}_j - \bar{q}$. Using the SVD, we can factor $Q = UDV^T$, where V is a 3×3 matrix containing the principal components of Q. The two largest principal components define the bases for the best fit plane; we are interested in extracting the normal, the smallest principal component, thus we are interested in the column of V corresponding to the smallest singular value.

A key observation here is that the computation of normals is extremely dependent on which points in the neighborhood of q_0 are chosen. In our case, we choose to use the

[3] our SVD implementation corresponds to their "PlanePCA" implementation

topology of the mesh to choose neighbors. However, as the resolution density of the scans increases, the normal estimates are increasingly sensitive to noise in the range returns. Thus, finding good normals may necessitate increasing the neighborhood size used in PlanePCA calculation.

One way to achieve this is to create an $O(n \log n)$ spatial data structure to find more points in the neighborhood. The extra computation is hard to justify, however, because a similar effect can be found by simply sub-sampling the laser data appropriately. The latter is the approach we choose here because it is very efficient and still produces good segmentations.

Once the normals have been estimated for each point for which color information is available, edge costs between points are straightforward to compute, using the angle between them. Let $\vec{n}_i$ and $\vec{n}_j$ be the unit normal vectors at two nodes i and j separated by an edge $e = (i, j)$, then the edge weight computed from the surface normals is given in Eq. 4.5:

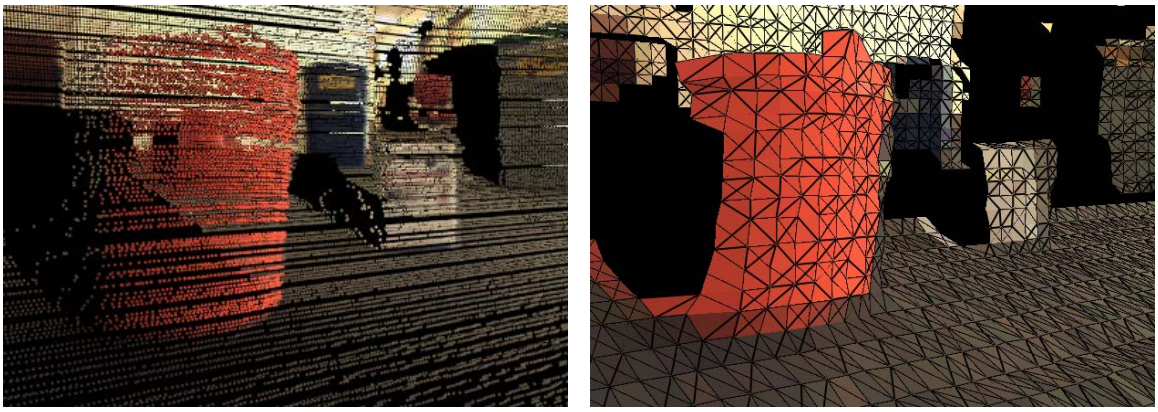
$$w_{norm} = \arccos\left(n_i \cdot n_j\right) \tag{4.5}$$

Algorithm 4.1 gives pseudocode for the complete segmentation pipeline using the mesh of colored points and the two edge costs from color and surface normals. The raw laser and camera data is first processed into an augmented set of point $p_i \in Points$, where $p_i = [x\ y\ z\ r\ g\ b\ n_x\ n_y\ n_z]$ and a set of edges $e_m \in Edges$, where $e_m = [i, j]$, where i, j are the indices of two points p_i and p_j. In lines 3 and 4, we compute edge weights, and sort the edges and the weights in increasing order. This takes $O(n + k)$ time. In line 5 we initialize a disjoint UnionFind data structure to keep track of which points belong to which segment [52]. Each union or find operation is $O\left(\alpha(n)\right)$, where α is the inverse Ackermann function, which is bounded by 5 for any practical value of n. Thus, a sequence of n unions or finds takes $O\left(\alpha(n)n\right)$, which is effectively a

Algorithm 4.1 SEGMENT_COLORED_POINT_CLOUD(L, I)

```
⟨Edges, Points⟩ = buildGraph(L, I)
Weights = computeEdgeWeights(Edges, Points)
countingSortOnRGB(Edges, Weights)
segments = new UnionFind(Edges)
init thresholds
for ⟨p_i, p_j⟩ ∈ Edges, ⟨w_rgb, w_norm⟩ ∈ Weights do
   S_i = segments.find(i)
   S_j = segments.find(j)
   if belowThresh(⟨w_rgb, w_norm, i, j⟩, thresholds) then
      newID = segments.join(i, j)
      updateThresh(⟨w_rgb, w_norm, newID⟩, thresholds)
   end if
end for
return segments
```

linear bound for any realistic number of nodes. In lines 7-9 we iterate through edges in increasing order of color weights, and propose unions between the sets connected by that edge. Line 10 shows that unions are only accepted if they meet both the dynamic criteria for both color edge weights and normal edge weights as described previously.



(a) Colored laser scan (b) Downsampled surface mesh

Figure 4.3: Spatially-derived segmentation mesh. We derived the spatially-informed mesh at a lower resolution to efficiently handle normal estimation using noisy LIDAR range returns.

4.3 Experimental Setup

To evaluate our method, we collected two distinct datasets, one indoors in an office hallway (the "hallway dataset"), and another outdoors in bright sunlight on a small hill (the "hill dataset"). Because our sensor configuration is novel, we are unable to evaluate our method using previously published datasets.

For our experiments, we used a Hokuyo UTM-30LX planar laser scanner, which returns 270 degree FOV planes at 40 Hz with 1/4° angular resolution. The laser is attached to a custom-made mount printed using rapid-prototyping ABS plastic, and actuated with an AX-12 Dynamixel servo (1/3° angular resolution). The camera is a color USB Firefly MV 03MTC camera with 752 x 480 resolution used in conjunction with a wide angle (111°) Tamron M13VM246 lens. The camera is attached to the same structure, and is actuated with a pan and tilt servo. For the data presented here, the camera is held fixed with respect to the robot; only the Hokuyo was actively actuated. This was to simplify the calibration process between the two sensors.

We analyzed the choices we made in designing our approach by comparing the performance with different aspects of the algorithm enabled or disabled. First, we

compared using a spatially-informed mesh, with a pixel-based grid for G. Second, we compared performance using different combinations of edge weights, either color-based, surface normal based, or both together.

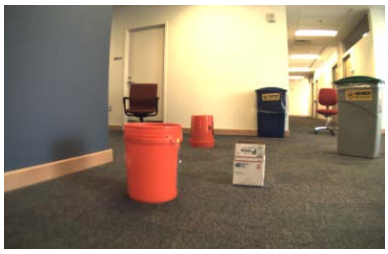
Our evaluation makes segment boundaries explicit by artificially coloring each segment where appropriate. Each dataset consists of a series of timestamped servo positions, camera images, and laser range measurements. For both datasets discussed below, we used similar parameters when possible, but varied to produce best-case results for each algorithm. The parameters we used are listed in Table 4.1 where 'Edge cut' denotes the maximum edge length threshold and both K_{rgb} and K_{norm} are the graph-cut parameters for their respective quantities. We also downsampled our LIDAR data uniformly, as noted in the table. This increased the fidelity of the PCA surface normal estimation algorithm, since small variations in surfaces were eliminated. A similar approach intended to smooth variation in surface normals is also employed in [51].

4.4 Results

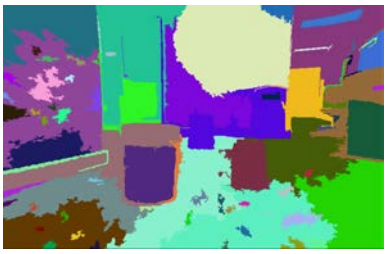
The hallway dataset has several everyday objects placed haphazardly to fill the scene. The camera image corresponding to this scene is shown in Figure 4.4a. We first show what previous methods from [36] produce using only the full resolution image. These results are shown in Figure 4.4b. Tuning the parameters for the image-only method can change which segments are created, but some objects cannot be

Figure	4.4c	4.4d	4.4e, 4.4f	4.4g, 4.4h	4.4j
Resolution (deg)	0.75	0.75	0.75	0.25	0.50
Edge cut (m)	0.25	0.25	0.25	0.25	0.50
K_{rgb}	1000	n/a	9000	9000	12000
K_{norm}	n/a	9	10	10	9
minsize	100	100	100	100	100

Table 4.1: Colored point cloud segmentation parameters

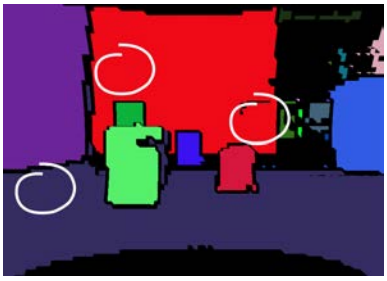


(a) The camera image for the "hallway" dataset.

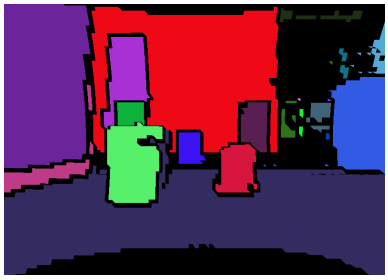


(b) Image-only segmentation from [36].

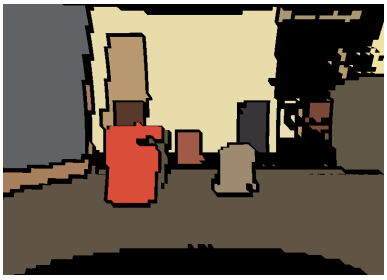
(c) Graph segments with using color only



(d) Graph segments with surface normals only



(e) Graph segments via our method.



(f) Our method (true color).

(g) Over-segmentation (true color).

(h) Alternate viewpoint of 4.4g.

(i) The camera image for the "hill" dataset.

(j) Graph segments (our method).

Figure 4.4: Segmentation evaluation. Our method differentiates on color and surface-normal changes. Note the undesirable results of other methods in 4.4c and 4.4d.

disambiguated on color alone.

This motivates the decision to use a spatially grounded mesh instead of a pixel-based mesh. Building the mesh from the laser scans also enables using additional information to suggest connections between regions in the scene. Using this mesh yields significant improvements over the vision-only method, even without any consideration of surface normals. This is shown in Figure 4.4c. However, as in the image-only segmentation, some of the objects cannot be disambiguated on color alone even when using the spatial mesh. In particular, the gray trash bin is difficult to separate from the floor using only color information.

We can also examine the segmentation performance when only surface normals are used to determine segment boundaries. Figure 4.4d shows how segmentation using surface normals is able to separate the gray trash barrel from the floor, but is unable to separate the blue recycling bin from the rear wall.

Our method combines both of these approaches to give a segmentation more consistent with our expectations. The results of the joint segmentation is shown in Figure 4.4e. For the segments using the recommended method, we also show how the segments look without false coloring; Figure 4.4f instead shows each segment colored by its average color.

Our method is considerably more robust than either laser data alone or color alone. To show this, consider the "hill dataset" scene depicted in Figure 4.4i. In this case, the sun is so bright that the camera (whose color calibration was fixed indoors) does a poor job of distinguishing between the color of the ground, the color of the orange buckets, and the color of the tree trunk. In this case, we observe that valid segmentation is still generated using the recommended method which determines segment breaks using both color and surface normals. This is shown in Figure 4.4j, where the barrels, the ground and the tree are segmented separately. Outdoor environments present a particular challenge for normal estimation algorithms because

terrain is more variable (e.g. grass), and consequently laser scanners produce results that are less consistent. By increasing k_{norm}, we are able produce a segmentation that separates the important objects in the scene. However, our metric for evaluating how "good" a segmentation remains subjective. Different values for the threshold constants will be required depending on the application (e.g. obstacle avoidance, or object detection). Finally, an example over-segmentation using our approach is shown in Figure 4.4g.

Even though the implementation we present is a batch operation on the data, the algorithm is fast enough to easily operate in real time. The scans in question were collected over a period of 12 seconds. The downsampled graph initialization over 148500 laser points took 176 milliseconds on the 2.6 GHz Core 2 Duo laptop attached to the robot. Segmentation of the downsampled 4161 points with color information took an additional 73 milliseconds. In this case, we are only using 2.1% of the CPU. (Note: even if the points are not downsampled, which produces poor segmentation due to range noise, the total runtime for segmenting 151201 points with color information is only 5.2 seconds, taking 43% of the CPU.) This suggests our method can be run in tandem with other processes that still use a significant amount of CPU power.

4.5 Discussion

Segmentation is an important pre-processing step necessary for higher-level tasks such as object recognition. We have demonstrated a novel method for efficiently and robustly segmenting a colored laser point cloud. Our method enables the detection of segment boundaries on the basis of both spatial cues and color information, improving performance in domains which previous unimodal methods found difficult.

Since objects often cannot be segmented (let alone recognized), without using color, our approach is crucial for boosting the capabilities of such higher level processes. Although our algorithm has a small number of tunable constants, we demon-

strated that our algorithm performs well in multiple environments using similar parameters.

This work predates the introduction of the Kinect by about 6 months. At the time of publication, our approach was the first segmentation algorithm applied directly to the RGB-D point cloud, likely a result of the difficulty of collecting carefully co-registered point clouds. The introduction of the Kinect, which co-located a calibrated range and color sensor, sparked a huge research effort in RGB-D pointcloud processing, leading to many more segmentation approaches and resulting in over 50 citations to this work, according to Google scholar. Subsequent works have leverage Moore's Law along with different computational-tradeoffs to achieve more robust results, for example, by using additional plane-fitting steps, or explicit removal of the ground plane [53, 54].

One of the main challenges which limits the impact of segmentation of the fused colored point cloud for teams of mobile robots is the difficulty of co-registering the camera and lidar data. Particularly for large teams of robots such as in the MAGIC competition, correctly calibrating and validating all the cameras in the fleet remained a prohibitive cost for our team. Therefore, during the actual competition, we performed segmentation separately in the point cloud and camera, and then fused the results later in the pipeline [2].

Achieving the potential of greater segmentation accuracy promised by the presented approach in urban reconnaissance requires a more streamlined approach to managing the calibrations of a fleet of robots. Recent work by Pandey et al. describes a method for automatic, targetless extrinsics calibration between laser scanners and cameras from real-world data [55]. Such an approach could be used to replace the manual correspondences we used for extrinsics calibration in this work. The remaining challenge of determining the camera intrinsics motivates the next chapter on robust, reliable and repeatable camera calibration.

CHAPTER 5

Guided Camera Calibration

Joint work with Andrew Richardson[1][2]

Applications such as colored point cloud segmentation [56], visual odometry [57] and dense reconstruction [58, 59] are fundamentally dependent on accurate calibrations to accurately combine camera data with other sensors readings, or to accurately extract metrical data from images. The MATLAB and OpenCV packages are two popular systems for calibrating lenses [60, 61][3]. These existing toolboxes can be error prone, however, especially for lenses with significant distortion. This stems from the fact that the quality of a calibration is dramatically affected by the user's choice of calibration images. A user who chooses poor calibration target positions may find the resulting model generalizes poorly to unseen examples. It can also be difficult or time consuming to independently verify that a calibration is correct. This is especially relevant to the deployment of large multi-robot teams, for example in the MAGIC competition. If additional data must be manually collected, processed, and inspected for each robot, it can quickly become prohibitive to rely on the calibrations to be accurate to the level needed for colored point cloud segmentation.

The calibration challenge is particularly acute for novice users, who are not aware

[1] Both Andrew and I contributed equally to this project

[2] ©2013 IEEE. Adapted with permission, from Andrew Richardson, Johannes Strom, Edwin Olson, "AprilCal: Assisted and repeatable camera calibration" September 2013.

[3] The more recent OpenCV calibration toolbox uses the same mathematical approach, and largely subsumes the functionality of the MATLAB toolbox, but with a more accessible user interface.

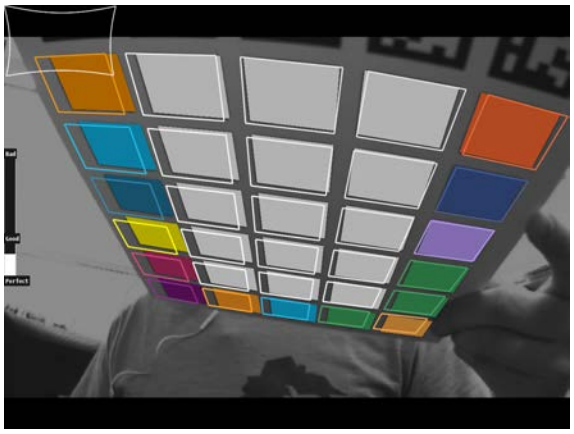


Figure 5.1: The AprilCal GUI. Our system combines the ability to reason about unseen targets and a novel quality metric to make suggestions to the user about where to place the target. The user is notified that calibration is complete once the desired accuracy has been reached, typically achieving < 1 pixel of error after 6-8 images.

of the properties of the underlying estimation and optimization methods, or end-users in dramatically different fields [62]. Even experts may be unsure that the target positions they have chosen will yield a sufficiently accurate calibration, as the number of images needed is not constant across lenses and should vary with the quality of the constraints. Consequently, standard practice is to collect many more images than necessary and verify that the model parameter uncertainty and training error are low; if the results are unsatisfactory, the calibration is repeated or updated with additional images. This process is unreliable, not theoretically grounded, and doesn't scale well to large numbers of robots or cameras.

The primary goal of this work is to increase calibration repeatability and accuracy in a more principled fashion. This work benefits both novice and expert practitioners in the field of robotics, machine vision, and downstream consumers of the results from these disciplines, for example, motion-capture systems which require precise camera calibration [63]. We introduce a paradigm where fit quality is explicitly considered at each stage during a *live* calibration process. Specifically, we automatically consider many unseen target positions and suggest positions that will best improve the

quality of the calibration. This is achieved using a novel quality metric based on the uncertainty of the calibration as measured in pixels. Previous toolboxes report the uncertainty of the *model* parameters, but the effect of these parameter uncertainties on pixel coordinates can be complex. We argue that worst-case uncertainty in pixels is more relevant for application performance and more natural for the user. Worst-case pixel uncertainty also serves as a principled basis to automatically determine when enough images have been collected.

We also introduce a new method for robustly bootstrapping a calibration that enables our system to make sensible recommendations even when little or no prior information is available about the lens. Our system also makes use of a calibration target composed of AprilTags [50], which, unlike previous approaches, can still be detected when individual markers are occluded. This enables a wider variety of target positions, which our method successfully exploits when making suggestions to the user.

We validated our camera calibration toolbox via a 16-participant study mostly compromised of users who had never calibrated a camera. Despite their lack of expertise, they were consistently able to use our software to produce accurate calibrations. The same novice users also used the OpenCV calibration toolbox and invariably produced poorer calibrations, in some cases yielding errors of tens of pixels. In addition, we show that our toolbox can calibrate a wide range of lenses with an explicit accuracy guarantee.

The contributions of this work include:

- A framework for generating target position suggestions to guide users during camera calibration
- A new evaluation metric for camera calibration that enables high confidence in calibration parameters everywhere in the image

- A bootstrapping setup to enable interactive calibration even before all parameters are fully constrained
- An evaluation that demonstrates the robustness of our approach using a human-subjects study of 16 people, mostly novices.

AprilCal is released as part of the APRIL Robotics Toolkit and is available at `http://april.eecs.umich.edu`

5.1 Background

A wide variety of camera calibration approaches exist, spanning different optimization methods, calibration target styles and intrinsic model designs. Many previous methods have used multiple views of a planar target [44, 60, 61] or a single view of a carefully constructed 3D target [64]. Other methods have used laser pointers or other bright lights to facilitate calibration of networks of cameras. Such approaches typically still require bootstrapping by calibrating some cameras in the network with a constructed target [65, 66]. All of these prior methods share the same approach to calibration: a user first collects a set of images, then runs a batch calibration process on that data. This is in contrast to our approach, where the entire calibration process is interactive and additional data is solicited until the desired accuracy has been achieved.

A dominant paradigm for calibration involves capturing several images of a planar target. These approaches (ours among them) make associations between points detected in the image and corresponding world points on the target whose relative position are known by construction [44]. Simultaneous optimization over the intrinsic parameters for the camera model and the extrinsics for each target yield an estimate of the model parameters. Using such an approach requires the choice of 1) optimization method and 2) lens model.

Among the many possible optimization techniques, we adopt a standard, iterative non-linear-least-squares approach, using a sparse matrix solver as the back-end. This method is roughly analogous to standard approaches in GraphSLAM and bundle adjustment [18, 67, 68]. Our calibration vector x consists of all the model parameters (roughly 10) for the camera, in addition to the 6-DOF position of each calibration target. For each image containing k extracted 2D image points, we add $2k$ linearized constraints as rows in the Jacobian matrix J. Each row-pair corresponds to projecting a feature from a known 3D coordinate on the calibration target into pixel coordinates, capturing both the unknown position of the camera and the unknown camera parameters. Iterative solutions to Eqn 5.1 yield a locally-optimal set of model parameters for x.

$$J^T \Sigma_z^{-1} J \Delta x = J^T \Sigma_z^{-1} r \tag{5.1}$$

$$x_{i+1} = x_i + \Delta x \tag{5.2}$$

Here, Σ_z is the matrix of prior covariances for the target detector, and r is the residual, the observed minus the predicted pixel coordinates for each point. The correct convergence of x to the global minimum is sensitive to initialization of x_0; we use a specific bootstrapping process to pick good initializations.

There are also a wide variety of models for camera intrinsics, starting with the fundamental pinhole model [69]. However, using the ideal pinhole model in isolation will poorly capture the dynamics of most real world lenses, especially those with a wide field of view. Therefore, many models extend this method by accounting for the lens distortion explicitly. For example, the MATLAB toolbox uses a polynomial Taylor series with 3-5 distortion terms to approximate these effects after projecting with the pinhole camera model [60]. In contrast, we have found that a polynomial as a function of θ, the angle from the principal axis, yields as good or better[4] calibrations,

[4]The details of this analysis are omitted due to space limitations. However, AprilCal

often with fewer distortion terms for the lenses tested, increases the stability of the calibration process, and handles $Z \leq 0$. This is a reduced version of the model by Kannala and Brandt [70], who also include tangential distortion. The specific preferred model for this work is described in Equations 3-8 in [71].

In a similar spirit to this work, several others have sought to make calibration easier, less time consuming, and less error prone. For example, the ROS calibration package for the PR2 now has specific guidelines for the user about which checkerboard positions are required for getting a "good" calibration [72]. However, even with good rules of thumb, it is still possible that a user will collect "bad" frames that lead to an inaccurate calibration. While it is possible to manipulate a calibration target automatically using a robotic arm [73], our approach can be used to choose desired target views during calibration. Others have shown that in some cases, it can be possible to calibrate a distorted camera using only a single image [62]. However, this approach does not explicitly constrain the accuracy of the resulting calibration, and works with only a very specific distortion model. In contrast, our method ensures that the user takes sufficient images to meet their desired level of accuracy, typically 6-8 images to achieve a < 1 pixel confidence.

Finally, as suggested in Ranganathan's work on non-parametric intrinsics models [16], we adopt the use of a strict testing set to provide a more rigorous evaluation of the actual accuracy of our method. This is in contrast to the standard practice, in this domain, of reporting only the *training* error.

5.2 Method

As outlined previously, our method improves on the state of the art by offering a virtual calibration assistant that provides suggestions to users and automatically

can perform model selection to evaluate all available models as a post-processing step. See `http://april.eecs.umich.edu` for details.

notifies them when the calibration has reached the specified accuracy. Our approach leverages a calibration target consisting of a mosaic of AprilTags [50], which can be detected robustly on the live video stream, even if parts of the target are occluded. Users interact with a GUI to match target positions suggested by our software. After each target position is achieved, the next best position is computed, repeating until the desired accuracy is achieved. To ensure a responsive UI, we use the aforementioned greedy strategy for choosing frames for the user to collect, rather than jointly reasoning about a sequence of future observations.

Our system divides the calibration into two sequential phases: bootstrapping, and uncertainty reduction. In the bootstrapping phase we start with a restricted camera model with very few parameters, relaxing to the full model as new images of the target are added. A detailed description of the bootstrapping process is covered in [71]. Once all model parameters are fully constrained (typically after 3 images), we switch to the uncertainty reduction phase until the desired accuracy is achieved. Transition from the first to second phase is transparent: in both cases, the UI remains the same.

5.2.1 Pixel-based Calibration Error Metric

Once all intrinsic parameters are fully constrained during the bootstrapping phase, the next goal is to guide the user to collect more images until the calibration accuracy requirements have been achieved. However, there are many possible images the user could collect, and we need a *metric* to rank which one will improve the accuracy the most. Previous approaches have used Mean Reprojection Error (MRE) and Mean Squared Error (MSE) as primary indicators of calibration quality. However, this is problematic because these are *training* errors, rather than testing errors. Using one of the prior works, if the training images are selected poorly, the resulting MRE could be low, yet the generalization performance (e.g. as measured on a test set) could

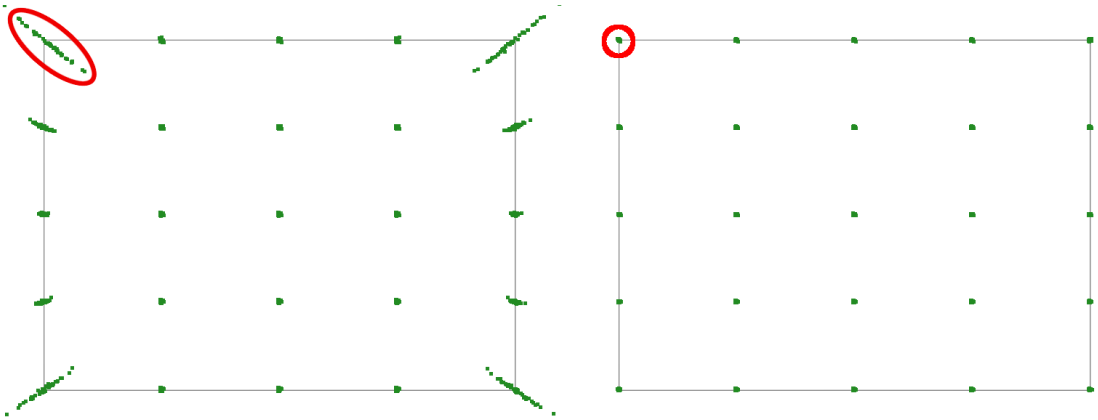


Figure 5.2: Max Expected Reprojection Error. The ERE for several test points is computed by sampling from the estimated model parameter distribution, and reprojecting world points to derive pixel-based error measurements. The resulting pixel error distributions are shown after 3 and 5 images have been taken by the user. The set of samples with the Max ERE are circled in red. Our sampling-based error metric can be used with any intrinsics model and allows us to automatically ensure the calibration is well constrained everywhere without requiring the user to collect a test set.

be very poor. Unfortunately, collecting a proper testing set can be onerous — for our evaluation we use an expert-selected set of 60 or more images from all over the camera's field of view. Especially for novice users, it is not reasonable to expect they would be able to collect a "good" testing set. Even for expert users, this process is time-consuming and requires careful attention.

Therefore, our approach is to derive a more principled estimate of the testing error that can be computed automatically given an intermediate state of the calibration. Our proposal is called "Max Expected Reprojection Error" (Max ERE), which we can compute at any stage during the calibration by sampling from the current posterior distribution over the model parameters. We then project a series of 3D points through each sampled calibration, producing a distribution of pixels for each test point, whose mean error is the "ERE". Finally, we take the max of the EREs over all the test points. This ensures that we will properly weight the part of the image where the model is currently the most uncertain. We choose the fixed 3D points carefully so that they

will project into all parts of the image. Our current implementation uses a 5 x 5 grid of test points distributed so their projections will uniformly cover the image (see Figure 5.2). See Alg. 5.1 for an overview of the implementation.

Algorithm 5.1 COMPUTE_MAX_ERE(currentCalibration)

```
(x, Σ) = getModelPosterior(currentCalibration)
meanCal = makeCal(x)
testPointsXYZ = makeTestGrid(meanCal, 5, 5)
calSamples = [sampleCal_0(x, Σ), ··· , sampleCal_n(x, Σ)]
MaxERE = 0
for all t⃗ ∈ testPointsXYZ do
   ERE = 0
   for all sampCal ∈ calSamples do
      ERE += (1/n)|meanCal.project(t⃗) - sampCal.project(t⃗)|
   end for
   MaxERE = max(ERE, MaxERE)
end for
return MaxERE
```

Computation of the Max ERE uses the estimate of the marginal posterior covariance of the model parameters: $\bar{P}(m|z_0, \cdots, z_n)$. This distribution is derived by first computing the joint distribution of the model parameters and each target extrinsics, given all the observations of those targets. Suppose we have collected n images of targets, then we can "marginalize-out" the target extrinsics:

$$\bar{P}\left(m|z_0, \cdots z_n\right) = \int_{T_0, \cdots, T_n} P\left(m, T_0, \cdots T_n|z_0, \cdots z_n\right) \tag{5.3}$$

where $m = \{f_x, f_y, \cdots, k_1, \cdots\}$, T_i is a 6DOF rigid-body transform, and z_i contains the detected x, y pixel locations of the centers of every AprilTag in image i. In practice, we assume the joint distribution in Eqn. 5.3 can be approximated as multi-

variate Gaussian:

$$N(x, \Sigma) = N\left(\begin{bmatrix} m \\ T_0 \\ \vdots \\ T_n \end{bmatrix}, \begin{bmatrix} \Sigma_{m,m} & \Sigma_{m,T_0} & \cdots & \Sigma_{m,T_n} \\ \Sigma_{T_0,m} & \ddots & & \vdots \\ \vdots & & \ddots & \\ \Sigma_{T_n,m} & \cdots & & \Sigma_{T_n,T_n} \end{bmatrix}\right) \tag{5.4}$$

This allows the marginal $P(m|z_0, \cdots) = N(m, \Sigma_{m,m})$ to be computed trivially by dropping the other rows and columns of the covariance matrix. Computation of $\Sigma_{m,m}$ requires forming the sparse, semi-positive definition, $\dim(x)$ by $\dim(x)$ information matrix $\mathcal{I}$, which is derived from the observed target positions (similar to Eqn 5.1):

$$\mathcal{I} = J^T \Sigma_z^{-1} J \tag{5.5}$$

where each row of J is the linearized projection equation describing how a point on the target projects into the image, given the model parameters m and target position T_i. $\Sigma_{m,m}$ can then be computed from the Cholesky decomposition of $\mathcal{I}$, as described in [74, 75].

Crucially, computing the information matrix depends on an estimate for the detector accuracy in pixels, σ_z, which must be known in advance. For AprilTag, we have empirically found the accuracy to be relatively constant across lenses with image width as a satisfactory predictor. Proper focus of the lens is assumed.

$$\sigma_z = 7 \times 10^{-5} \times \text{width} \tag{5.6}$$

Detector accuracy was fit independently for a number of camera configurations (see Figure 5.8) using 60+ image calibration datasets for each. The resulting accuracies were then used to compute the linear model in Eqn. 5.6.

Using the derived, low-dimensional marginal distribution over the model parameters, $P(m|z_0, \cdots)$, we can then sample a series of m_i from P to compute the Max Expected Reprojection Error, as described above. This metric is then used to select the next target the user is guided to achieve.

5.3 Implementation

AprilCal is implemented in Java and runs at 25 FPS with 640×480 images on a quad-core Intel i7-3740QM @ 2.7GHz. Using a mosaic of AprilTags as our calibration target allows us to automatically detect the target at video rates, with processing time typically dominated by AprilTag detection. We have found that rigid mounting at an office supply store is inexpensive and yields a target durable enough for many uses. In addition, we can detect and recognize individual tags without observing the entire target. This makes it possible to add constraints in the corners of images, even for highly distorted lenses. In the multi-camera case, this allows for the calibration of cameras with adjacent, but non-overlapping, fields of view.

In addition to target detection, our implementation requires significant CPU when determining the next suggestion. In our implementation, we score a fixed set of about 60 target positions regularly distributed throughout the view frustrum. This process depends on incorporating hypothetical observations into the calibration optimization framework, and then estimating the marginal distribution over the model parameters. As more images are acquired, and the size of the joint distribution grows, this can take up to 1 or 2 seconds. However, this scoring process only occurs a small number of times: once after each suggestion has been achieved by the user.

The AprilCal user interface is designed primarily to allow the user to correctly match the target positions suggested by the system. As shown in Figure 5.1, we use a set of unique colors to show how the desired target position (hollow rectangles) should be matched by the live detections (solid rectangles). The UI also automati-

cally offers basic advice to the user via textual prompts about how to move the target to match the desired pose. When the calibration is deemed complete, the user is then automatically presented with the rectified video stream. This allows the user to qualitatively verify that the calibration is accurate, primarily by checking for straightness of projected lines.

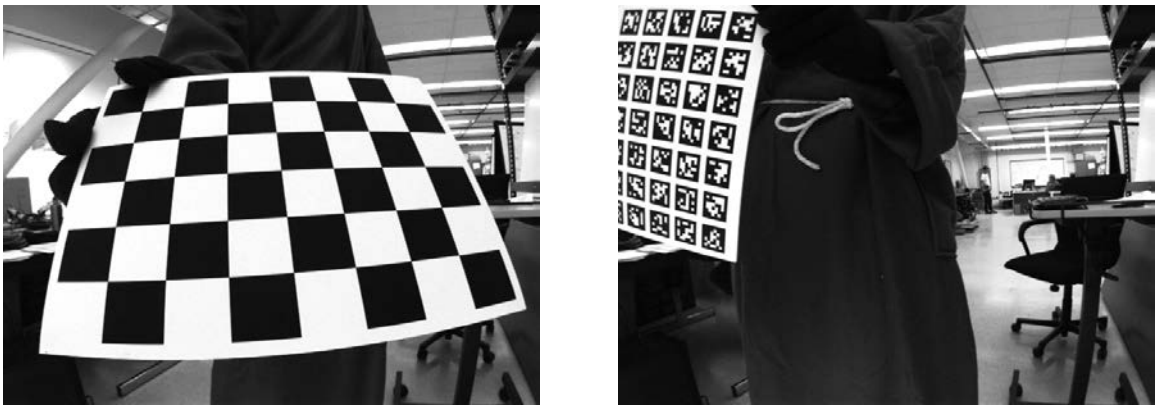
5.4 Human trials

We conducted a series tests with human subjects[5] to measure the effectiveness of AprilCal and to compare it to the widely used OpenCV method. Our user population consisted of undergraduate students at the University of Michigan. Only 3 of the 16 subjects reported any previous experience with camera calibration.

Each participant was asked to calibrate the same camera and medium-distortion lens with two different methods (see Figure 5.3). We used a Point Grey Chameleon CMLN-13S2M-CS in 648×482 8-bit grayscale mode with a 2.8mm Tamron lens (Model 13FM28IR). This lens has a medium amount of distortion — significant enough that several Taylor series terms are required to model it, but still with a moderate field of view (only 93° horizontal FOV).

The two methods we evaluated were OpenCV's calibration using automatic checkerboard detection and AprilCal using a mosaic of AprilTags. Participants were given a set of printed instructions. If they asked questions to the experimenter, they were given comprehension-level clarification on the instructions or advised to re-read the instructions. Participants then followed a checklist to first collect four samples using OpenCV, followed by four samples using AprilCal. Additionally, participants watched a video demonstrating calibration with each method. In Method A (OpenCV), participants interacted with a GUI showing live detections of the chessboard, using the

[5]Our study was reviewed by the University of Michigan Humanities Institutional Review Board and designated "exempt" with oversight number HUM00066852.



(a) (b)

Figure 5.3: Human study sample images. Participants took images of a checkboard for Method A (Open CV) and an AprilTag mosaic for Method B (AprilCal). A smock and gloves were used to protect of subject anonymity.

"space" key on a wireless keyboard to capture a frame. In Method B (ApriCal), the frames are automatically taken when the participants move the targets close enough to the suggested pose.

In contrast to AprilCal, which provides detailed guidance throughout the calibration, OpenCV's `calibrate.cpp` provides no in-application suggestions. To enable a fair comparison, we provided a set of written instructions for calibrating with their software. Our goal was primarily to emulate the experience of a first-time user who downloads this software from the Internet. Therefore, we provided users with some example pictures from the MATLAB Toolbox web page. The best written instructions we found were on the ROS tutorial for monocular camera calibration [72], which we also included. These are:

- checkerboard on the camera's left, right top and bottom of field of view
- checkerboard at various sizes
- close (fill the whole view)
- far (fill $\sim$1/5 of the view)

Dataset	Lens Model	Reprojection Error	
		Mean	Max
OpenCV	Radial, 3 distortion terms	0.728	38.646
AprilCal	Radial, 3 distortion terms	0.229	1.651
AprilCal	Angular, 4 distortion terms	0.203	1.444

Table 5.1: Human study calibration performance evaluation. Average mean and max testing errors show AprilCal calibrations are lower error, as measured over all human subject trials against a a 65 image reference dataset. Results are significant with $p < 6.3 \times 10^{-7}$ for both mean and max errors and all pairs of rows.

- checkerboard tilted to the left, right top and bottom

After reading these instructions, participants were then instructed to take 10-16 images in each of the 4 trials (on the same lens).

5.5 Evaluation

We evaluated AprilCal on several fronts. First, we report the results from our human subjects study to demonstrate the robustness of our approach to user error. Then, we demonstrate that our novel Max ERE metric is a good proxy for testing error. Finally, we demonstrate that AprilCal can be successfully used on a variety of lenses.

A note on evaluation of calibration quality: in all of our evaluations, we use testing error to indicate calibration quality. Each testing set is a collection of 60+ images from all over the field of view, including the corners of the images and at various scales. Because we do not have ground truth positions for the targets in the testing set, the error we report is after we optimize the target extrinsics to best fit the **fixed** model parameters for a given calibration. While this in general results in lower reprojection errors, it ensures that all models are fairly evaluated and still allows discrimination between good and bad calibrations.

Furthermore, Mean Reprojection Error (MRE) is typically reported as a summary of calibration quality, as it is simple to understand and robust to detector error. How-

ever, it can also often mask systematic errors in the underlying calibration. Therefore, we also report Max Reprojection Error on the test set — this ensures that calibrations are evaluated by their performance everywhere in the image.

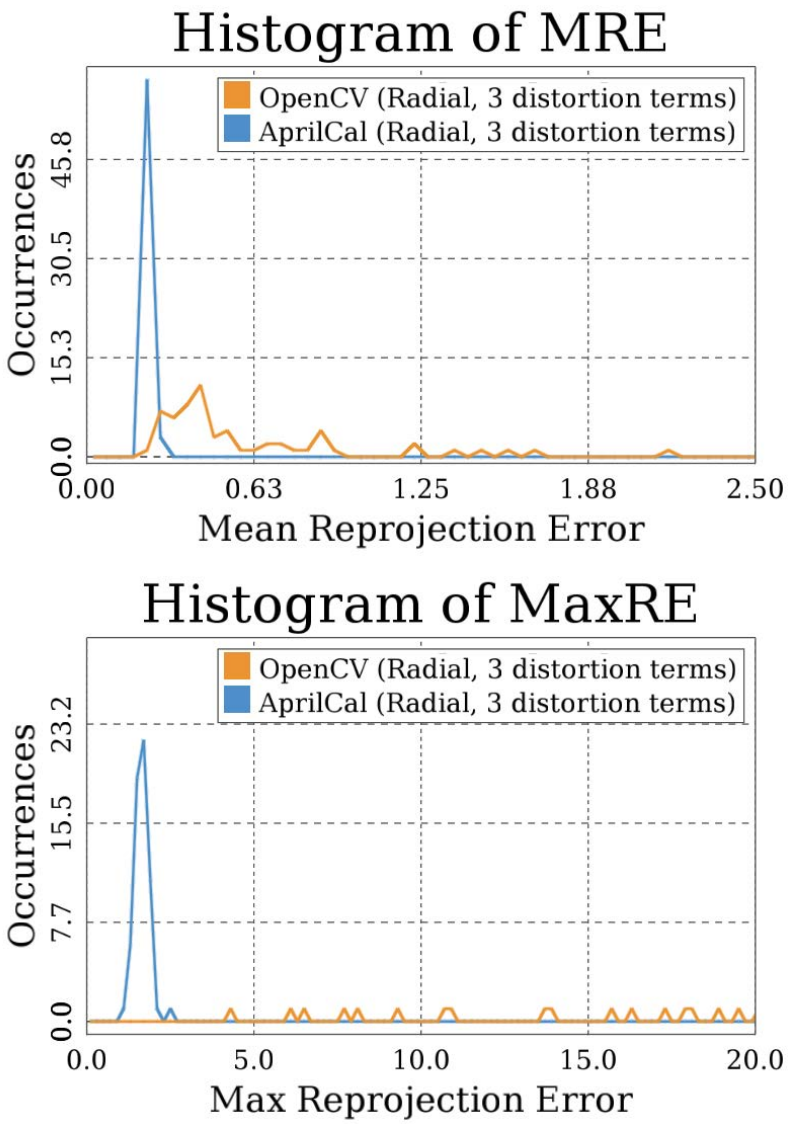
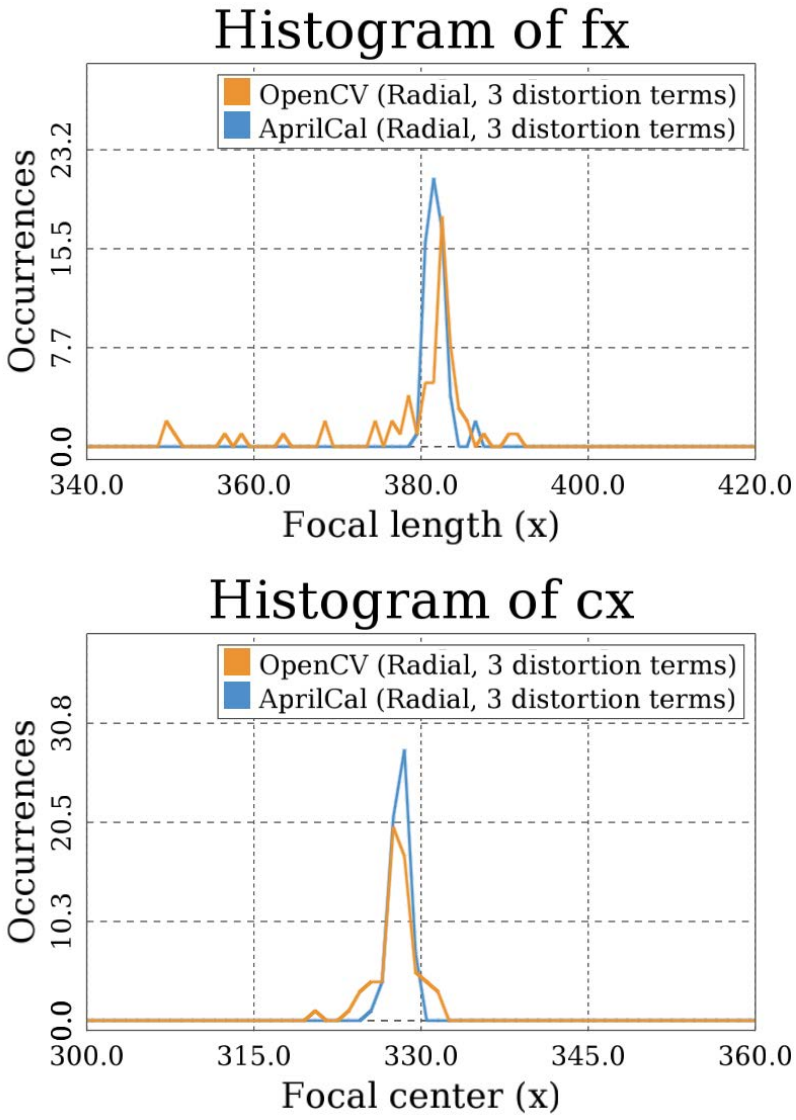


Figure 5.4: Human study error histograms for mean and max reprojection error. Errors are computed with respect to a 65-image test set for calibrations produced by the participants using AprilCal and OpenCV. For OpenCV, 3 MREs and 46 MaxREs not visible within plot extents.



Dataset	Focal length (x)		Focal center (x)	
	Mean	**Std dev**	**Mean**	**Std dev**
OpenCV	378.9†	9.0†	327.8†	1.9†
AprilCal	381.7	1.2	328.0	0.8

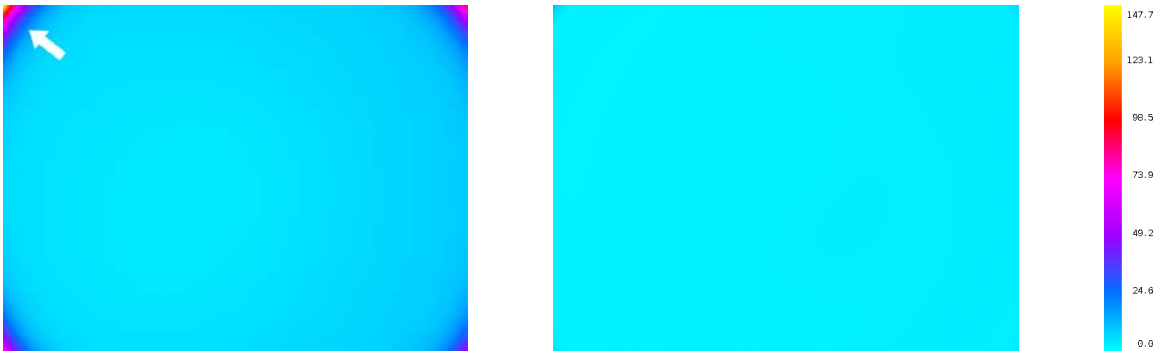
Figure 5.5: Human study focal- length and center distributions. While the mean parameter values from calibrations with AprilCal and OpenCV are similar, the standard deviations for the OpenCV calibrations are much higher. †One OpenCV outlier that would have further skewed the calculations was omitted from the calculations and is not visible within the plot extents. Best viewed in color.

5.5.1 Novice Calibrators using AprilCal and OpenCV

Our study results show that novices do a significantly better job calibrating when using AprilCal than when using OpenCV ($p < 6.3 \times 10^{-7}$). For example, with testing set errors averaged over all participants, the testing MRE using OpenCV is approximately three times that when using AprilCal (see Table 5.1). The disparity is even greater when considering the Max Reprojection Errors — OpenCV *averages* 38 pixels (6% of the image), whereas AprilCal averages a much lower 1.6 pixels for the same model. Interestingly, no OpenCV calibration yielded a max reprojection error better than the *worst* max reprojection error from AprilCal (2.02 pixels). This may be because the sorts of images that novice users capture, even when attempting to follow the ROS instructions, do not constrain the model parameters well. With the target suggestions provided by AprilCal, even new users of camera calibration software can produce calibrations with very low worst-case reprojection errors. The error histograms for both populations is shown in Figure 5.4.

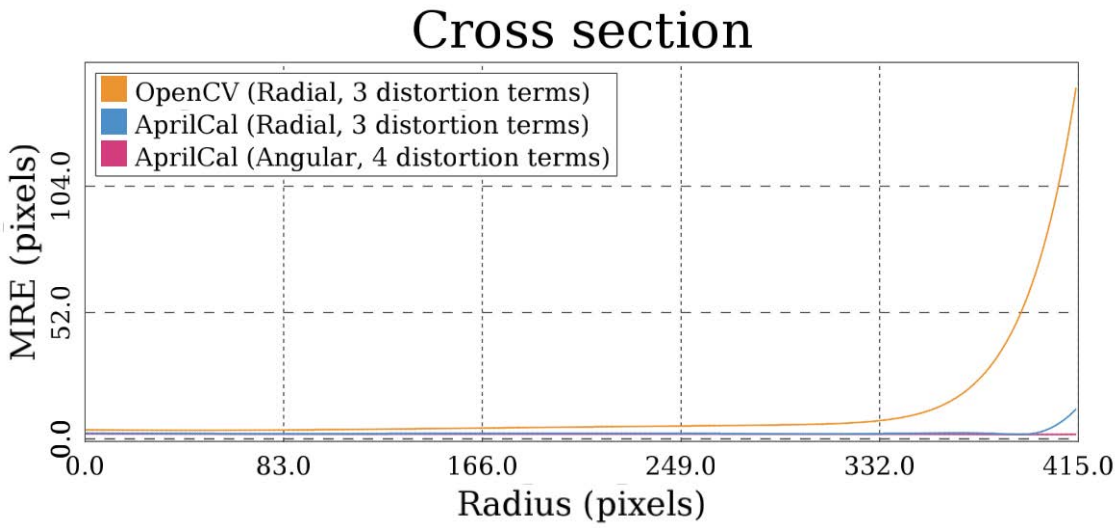
The human study results can also help us understand where in the image the calibrations disagree. Figure 5.6 depicts the expected error between the human trials and a 65 image reference calibration. From the images shown, it is clear that the OpenCV calibrations fail to capture the lens model in the image corners. This can be explained by both the need to observe the whole calibration target in OpenCV and the difficulty for users to predict where constraints are needed.

In addition to showing that AprilCal calibrations are more accurate, the user study results also show that calibrations with AprilCal are more consistent. Figure 5.5 depicts the distribution of focal lengths and focal centers for both AprilCal and OpenCV. While both distributions have similar means, the AprilCal standard deviations are 7.5× smaller for focal lengths and 2.37× smaller for focal centers.



(a) OpenCV (Radial, 3 dist. terms)

(b) AprilCal (Radial, 3 dist. terms)



(d) Cross section from the focal center to the furthest corner in the image

Figure 5.6: Reprojection error heatmaps for AprilCal and OpenCV. Reprojection errors were calculated by projecting a point for every pixel in the reference calibration through all test subject calibrations, then computing the pixel distance. The reference calibration used an angular polynomial model with 4 terms, as it had the lowest mean and max training errors. AprilCal calibrations show low error in all parts of the image, while OpenCV calibrations have very high error in the corners (see arrow).

5.5.2 Evaluating the Max ERE metric

We designed the Max ERE to be a good measure of calibration quality. Specifically, we want users to specify the accuracy they need for their application (e.g. $<$ 1 px), and if the Max ERE falls below that threshold, then the calibration can confidently be said to be that accurate. To validate these claims, we computed several variants of testing error over a large number of AprilCal trials. After each image is added, we evaluate the performance using Max ERE, as well as on an independent testing set using Max (100th percentile), 99.5th percentile, and mean reprojection errors. As can be seen in Figure 5.7, our sampled-MRE metric corresponds closely to the highest percentiles of testing error. This shows empirical evidence that our metric is effective.

5.5.3 Accuracy of AprilCal on a Variety of Lenses

In addition to performing reliably for a wide range of users, AprilCal also produces accurate calibrations for a number of camera and lens configurations. Each lens was calibrated multiple times by one of the authors using the guidance provided by AprilCal (typically requiring 6-8 images in total). A separate 60+ image testing set was collected to evaluate the accuracy for each configuration. To fairly compare results from different lenses, we compute each lens' testing error against a reference calibration computed from the corresponding test set. This eliminates the effects of detector error on testing error, which varies for different images sizes (see Eqn 5.6). For each target point detected in the testing set, we project through both the reference calibration and the calibration using AprilCal, and compute the Mean, 99.5^{th} percentile and Max Reprojections Errors. Figure 5.8 shows the testing error for six configurations that our lab uses for various robotics applications, including stereo odometry, object detection and overhead ground truth. In each case, the testing MRE is significantly below one pixel. Example images from each configuration are shown in Figure 5.9.

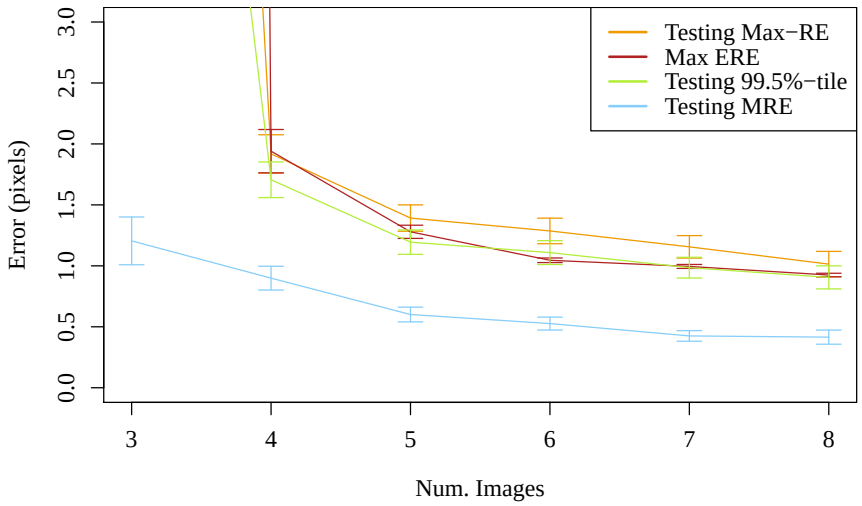
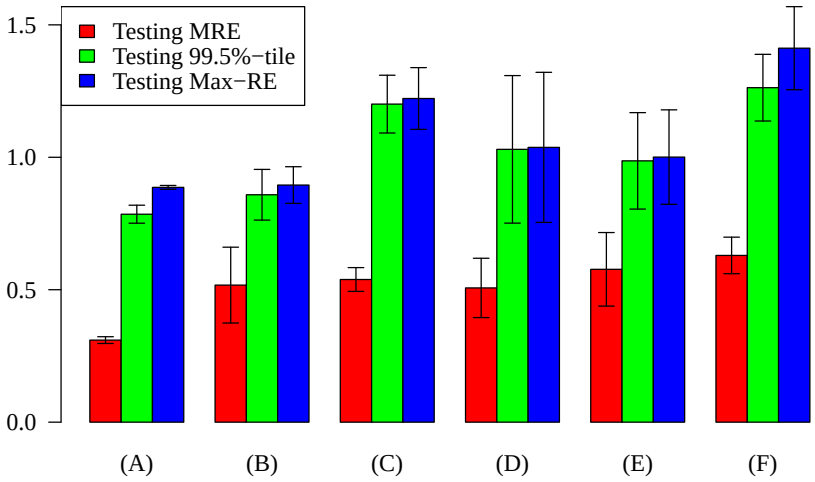
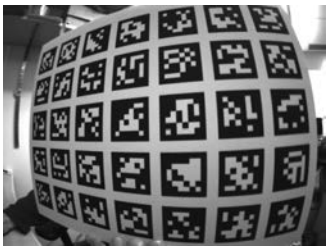


Figure 5.7: Max ERE correlation with testing error. Our novel Max ERE metric (red) correlates highly with the 99.5th percentile of reprojection errors on an independently captured test set. The Max ERE allows us to automatically compute a rigorous accuracy score for a partial calibration without needing an exhaustive test set. Error bars reflect std. error of the mean.

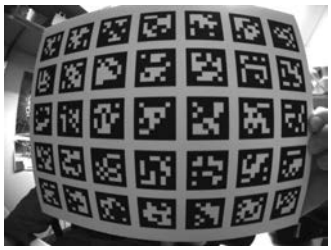


	Lens	DFOV	Resolution	Format
(a)	Fujinon YV2.2x1.4A-2	143°	648 × 482	Gray
(b)	Tamron 13FM22IR	146°	648 × 482	Gray
(c)	Tamron 13FM28IR	114°	648 × 482	Gray
(d)	Boowon BW38B	83°	752 × 480	Color
(e)	Boowon BW3M30B	121°	648 × 482	Gray
(f)	Boowon BW3M30B	121°	1296 × 964	Color

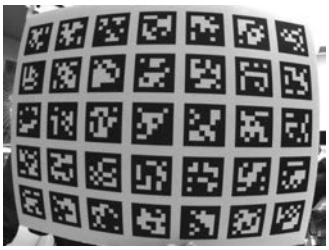
Figure 5.8: AprilCal evaluated on multiple lenses. AprilCal allows accurate calibration of a variety of lenses, as measured by error against a rigorous test set. The Diagonal Field Of View (DFOV) was estimated from the testing sets for each configuration. Error bars reflect std. error of the mean.



(a) Fujinon YV2.2x1.4A-2 (648 × 482)

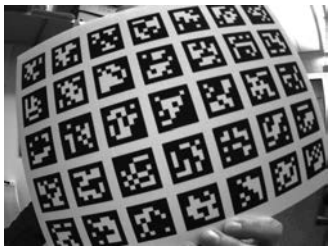


(b) Tamron 13FM22IR (648 × 482)

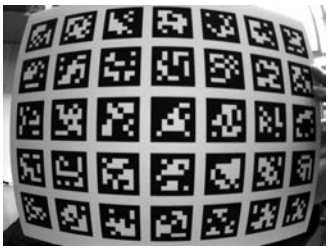


(c) Tamron 13FM28IR (648 × 482)

(d) Boowon BW38B (752 × 480)



(e) Boowon BW3M30B (648 × 482)



(f) Boowon BW3M30B (1296 × 964)

Figure 5.9: Sample images for each lens used in the evaluation. AprilCal is demonstrated on a variety of lenses with moderate amounts of radial distortion. Image contrast adjusted for clarity.

5.6 Discussion

AprilCal is an interactive calibration tool that provides live feedback on the state of the calibration and produces tightly-distributed calibration parameters even when used by novices. This problem is challenging to solve because the correct lens parameters are not know initially, so the system must make recommendations under significant uncertainty. We address this challenge by leveraged a novel calibration quality metric (Max ERE), which we derived from the observed uncertainty distribution over the intrinsics parameters. We leverage Max ERE to automatically determine whether a calibration is sufficiently accurate, without requiring a user to collect a rigorous testing set. We conducted a 16-person human subjects study to show that even novice users can produce consistent, quality calibrations using such a system.

AprilCal is especially useful when many cameras need to be calibrated, for example for a team of urban reconnaissance robots. Using AprilCal allows the fleet

maintainer to calibrate each camera and confidently use the resulting calibrations without needing to carefully validate the parameters on each robot. Furthermore, the Max ERE metric we presented could be adapted to allow robots to calibrate their cameras automatically, for example if a target is mounted on a mechanical arm that is wirelessly controlled by the robot.

CHAPTER 6

Conclusion

Real-world deployments of multi-robot teams face a number of challenges, including mechanical robustness, effective collaboration under uncertainty, human-robot interfaces, and understanding raw sensor data. In this thesis, we have addressed some of these challenges by demonstrating a number of online perception and mapping algorithms that scale to teams of robots cooperatively mapping and exploring urban environments. These algorithms are fundamental building blocks that higher-level planning and object-detection algorithms can use to understand and interact with large, real-world, indoor-outdoor urban environments. During the MAGIC 2010 competition, the APRIL lab fielded a vertically integrated multi-robot exploration team (including both hardware and software design). The work presented here either contributed to our success, or was motivated by challenges we experienced during our participation. While our approaches are tuned for MAGIC-like environments, these algorithms (perhaps with different parameter tunings) are still useful more broadly in the search and rescue, SLAM, and machine vision communities, among others.

When we started designing the MAGIC team, we quickly identified state estimation and mapping as one of the core challenges, as the competition venue required robots to operate in *a priori* unknown environments. The SLAM field had been focused on this challenge for several years before MAGIC, and while many aspects of

SLAM have been solved, the application to autonomous multi-robot teams lagged behind. The key challenges when applying SLAM to multi-robot teams included automatic loop-closure identification, and generating large maps of all the sensor data. The automatic loop-closure challenge, addressed by Olson in [14], involved scaling SLAM approaches to many robots by making significant algorithmic improvements to scan matching by introducing a heap-based multi-resolution search. These loop closures form the backbone of the multi-robot SLAM graph. The challenge of rasterizing gridmaps from the SLAM graph online is addressed in this thesis. Unlike existing SLAM approaches that focus on single-robot operation, we needed to develop a strategy to handle data from 14 robots at once, and still be able to fuse it to produce an accurate state estimate. The covering-based decimation, along with an efficient, grid-aligned caching data structure, were the key insights that enabled us to improve the efficiency of this process by avoiding the need to reprocess all historical data at every time step.

In contrast to the parameteric SLAM approaches we fielded for MAGIC, occupancy-grid mapping has largely been used in the SLAM community in the context of sampling-based inference methods, such as FastSLAM [8]. These approaches maintain a rasterized representation of historical data, and associate it with a large number of particle hypotheses. Due to the particle depletion these approaches suffer in high-dimensional state spaces, they are poorly suited for inference over an entire multi-robot team. Therefore, one key lesson learned from the rasterization methods presented here is that it is possible to use occupancy grids efficiently in the context of parameteric SLAM approaches. The rasterization implementation presented here is designed for execution by a CPU, but many of the operations are easily parallelizable. Future work is needed to determine whether GPU memory access speeds are fast enough to allow an overall speedup. Looking forward, variants of the caching and sparsification techniques here could also be applied in other areas, for example, to the

problem of generating city- or country-scale rasterized maps to support autonomous vehicles on public road infrastructure. In particular, such maps must be built incrementally, and it is important to be able to efficiently update them, for example when lanes are moved around a construction zone.

The CoverComposite algorithm improves over the strictly-linear performance of the naïve algorithms, but it still has worst-case linear time, for example in the case where robots never revisit previously mapped areas. For scaling to such environments, the SLAM estimation problem, as well as rasterization, will need to be able to "freeze" certain parts of the map in order to work within a fixed compute capacity, such as in the Atlas framework [76].

The MAGIC competition also highlighted the need for high-bandwidth, reliable communication infrastructure. However, the urban environments where such teams are designed to operate are very challenging from a signal-propagation perspective. Existing cellular networks cannot always be relied on, so multi-robot teams must leverage their capability to deploy their own infrastructure. High-bandwidth, short-medium range radios, such as 802.11, are ideal for this due to their portability, low-cost, and use of the ISM frequency bands. However, such radios would need to be deployed as a mobile mesh-network, which introduces a key coupling between how robots decide to move (to maintain the mesh), and the signal-propagation properties of their environment. This is especially challenging due to the fact that environments are *a priori* unknown, meaning that robots must collect data about their surroundings and react to it dynamically.

In Chapter 3, we undertook the first half of this challenge: given that robots maintain a mesh network, and collect signal strength and LIDAR data, we must predict where they will be able to communicate in the future. Therefore, we developed MATTE, a tomographic approach which fuses LIDAR and signal strength estimates to help make accurate predictions of signal strength. This problem was very chal-

lenging to solve, because the signal strength measurements are very sparse, biased, and one-sided: they only cover the links where a pair of robots were, at a particular instant in time, and furthermore, the signal strength is only measured when a packet is successfully received. Nonetheless, we showed that the approach of combining additional data from the LIDAR to help predict signal strength is promising. One important contribution of the MATTE work is the large amount of real-world data we collected to validate the approach. This was several orders of magnitude bigger than the largest existing evaluation in the signal strength prediction literature.

One particular environmental aspect we didn't address directly in MATTE is how knowledge of elevation, such as hills or ridges could be taken into account. The environments we tested were predominantly planar. Nonetheless, it should be possible to leverage 3D point cloud data to compute elevation, or to use pre-existing GIS databases, to help improve the prediction quality. Additionally, another application of our approach could be in fusing signal strength measurements with satellite imagery. It may be possible to automatically classify the imagery into categories (such as road, trees, buildings) and learn spatially varying attenuation rates for each category separately, similar to how the MATTE algorithm does this for LIDAR-estimated classifications of structure, free-space, and unknown.

The second half of the communications challenge for cooperative multi-robot teams is to develop coordination strategies under joint communications and environmental structure uncertainty. Reasoning under uncertainty is computationally challenging, and is exacerbated by the number of robots with uncertain state. In some follow on work to MATTE, we explored some of these issues, by allowing agents to model each other, and their future actions. However, tackling this problem in interesting, uncertain environments remains a challenging topic. Some of our efforts in this direction are discussed in a separate tech report [77]. Others in the APRIL lab are studying similar approaches to planning under uncertainty, using compact

models of agents, in the domains of navigation, and autonomous vehicle decision making [78, 79].

One important takeaway from the MATTE work is that fusing together sensors with different characteristics can help imbue robots with a richer understanding of their surroundings. Similarly, in the colored point-cloud segmentation work detailed in Chapter 4, we achieve improved segmentation by fusing together data from 3D LIDAR and a camera. It is not surprising that improved quality can be obtained by using more data. However, efficiently processing more data is challenging. Our approach addresses efficiency concerns with near-linear complexity. Since segmentation is just one part of an object recognition pipeline, it is important to use as little CPU as possible. Therefore we demonstrate how our approach achieves subjectively good segmentation, but using only 5% CPU. While our work is the first online segmentation algorithm applied to colored point clouds, subsequent work has built more complex processing pipelines with worse complexity bounds to achieve more robust performance [53, 54]. In this sense, our work helps to set the baseline for what is achievable using linear complexity so future approaches can show their additional computation is warranted.

There remain acute challenges in the segmentation domain, however, including establishing quantitative evaluation metrics and the necessity of carefully co-calibrating sensors. Quantitative metrics are difficult to employ because it is the end-to-end performance of an object detection pipeline that is of most concern. Depending on the other stages of the pipeline, different performance of the segmentation step may be required. While some subsequent work has evaluated their approach using ground-truth segmentation as specified by a human [54], future work in this area should evaluate the segmentation process in the context of the final application to give meaningful quantitative evaluations.

The calibration challenges in our segmentation work motivated Chapter 5 on ro-

bust and repeatable camera calibration. Maintaining the camera calibrations on the MAGIC fleet of 14 robots using the standard calibration approach at the time proved impractical; the MATLAB toolbox (predecessor of the OpenCV calibrator) was time consuming, required batch processing, needed manual validation of the checkerboard detections, and was error-prone, since novice users were free to choose arbitrarily bad calibration images. Validating a calibration was also not directly part of the MATLAB toolbox workflow, which relied instead on the user to design their own time-consuming experiments to determine the quality of the estimated calibration parameters.

Our software, AprilCal, is a significant departure from this methodology. We developed an online, interactive, and repeatable calibration technique, grounded in principled and automatic analysis of the model parameter uncertainties to guide users to pick the right exemplar images to properly constrain all parts of the camera model. Using this software, it is much easier to calibrate a fleet of robot sensors — an experienced AprilCal user can finish a calibration in a few minutes. The robustness of the approach eliminates the need for secondary validation, thereby making it easier to recalibrate in the field, or any time the configuration of the cameras is changed.

Future work in this area should consider how robots with actuators can be used to automatically reposition a camera to collect the desired images. Furthermore, it is possible that a similar approach to reasoning about the intrinsics uncertainty could be used to inform which data to collect for repeatable extrinsics calibration between the camera and LIDAR. Especially if integrated into an automatic process, this would improve the applicability of the presented color point cloud segmentation algorithm to field robots. AprilCal also has the potential to impact communities far outside the robotics domain, including machine vision as well as any inter-disciplinary study where researchers from other fields are looking to use calibrated cameras for data collection.

These contributions are all related by the multi-robot applications which inspired their development, but they also leverage similar grid-based or multi-variate-Gaussian inference methods. Many of the methods also rely on carefully exploiting the sparse structure of the estimation technique to achieve sufficient efficiency gains so that the solutions can be run online, on real robots. All of our methods are evaluated on real data collected in a variety of settings, including large urban environments and human-subject trials. By demonstrating the value of our approaches on real-world data, we make a more convincing argument for the adoption of these algorithms in the robotics community, and beyond.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] S. Balakirsky, S. Carpin, A. Kleiner, M. Lewis, A. Visser, J. Wang, and V. A. Ziparo, "Towards heterogeneous robot teams for disaster mitigation: Results and performance metrics from robocup rescue," *Journal of Field Robotics*, vol. 24, no. 11-12, pp. 943–967, 2007.

[2] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, and M. Bulic, "Progress towards multi-robot reconnaissance and the magic 2010 competition," *Journal of Field Robotics*, vol. 29, pp. 762–792, September 2012.

[3] G. Hollinger, S. Singh, and A. Kehagias, "Efficient, guaranteed search with multi-agent teams," in *Robotics: Science and Systems*, Washington, D.C., 2009.

[4] R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. Erkmen, "Search and rescue robotics," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer Berlin Heidelberg, 2008, pp. 1151–1173.

[5] S. Kawatsuma, M. Fukushima, and T. Okada, "Emergency response by robots to Fukushima-Daiichi accident: summary and lessons learned," *Industrial Robot: An International Journal*, vol. 39, no. 5, pp. 428–435, 2012.

[6] S. E. Vozar, "A framework for improving the speed and performance of teleoperated mobile manipulators," Ph.D. dissertation, The University of Michigan, 2013.

[7] R. D'Andrea and P. Wurman, "Future challenges of coordinating hundreds of autonomous vehicles in distribution facilities," in *Technologies for Practical Robot Applications, 2008. TePRA 2008. IEEE International Conference on*, Woburn, MA, Nov 2008, pp. 80–83.

[8] M. Montemerlo, "FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[9] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 143–156.

[10] P. Ranganathan, R. Morton, A. Richardson, J. Strom, R. Goeddel, M. Bulic, and E. Olson, "Coordinating a team of robots for urban reconnaisance," in *Proceedings of the Land Warfare Conference (LWC)*, November 2010.

[11] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proceedings of the second international conference on Autonomous agents*, ser. AGENTS '98. New York, NY, USA: ACM, 1998, pp. 47–53.

[12] S. Thrun and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures," *International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–430, May-June 2006.

[13] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Rome; Italy, April 2007, pp. 1670–1677.

[14] E. Olson, "M3RSM: Many-to-many multi-resolution scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seatle, May 2015.

[15] J. Fink, A. Ribeiro, and V. Kumar, "Motion planning for robust wireless networking," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. St. Paul: IEEE, 2012, pp. 2419–2426.

[16] P. Ranganathan and E. Olson, "Gaussian process for lens distortion modeling," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012.

[17] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of AAAI-00*, Austin, 2000.

[18] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, December 2006.

[19] E. Olson, J. Leonard, and S. Teller, "Spatially-adaptive learning rates for online incremental SLAM," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[20] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.

[21] B. Balaguer, S. Balakirsky, S. Carpin, and A. Visser, "Evaluating maps produced by urban search and rescue robots: Lessons learned from RoboCup," *Autonomous Robotics*, vol. 27, pp. 449–464, 2009.

[22] J. Strom and E. Olson, "Multi-sensor ATTenuation Estimation (MATTE): Signal-strength prediction for teams of robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, October 2012.

[23] A. Ghaffarkhah and Y. Mostofi, "Channel learning and communication-aware motion planning in mobile networks," in *American Control Conference (ACC)*, Baltimore, July 2010, pp. 5413–5420.

[24] J. Wilson and N. Patwari, "Radio tomographic imaging with wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 9, no. 5, pp. 621–632, May 2010.

[25] M. Malmirchegini and Y. Mostofi, "On the spatial predictability of communication channels," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 3, pp. 964–978, march 2012.

[26] N. Michael, M. Zavlanos, V. Kumar, and G. Pappas, "Maintaining connectivity in mobile robot networks," *Experimental Robotics*, pp. 117–126, 2009.

[27] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2005.

[28] H. Zhang, M. Genton, and P. Liu, "Compactly supported radial basis function kernels," North Carlolina State, Tech. Rep., 2004.

[29] J. Strom and E. Olson, "Occupancy grid rasterization in large environments for teams of robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, 2011.

[30] E. Olson, "Real-time correlative scan matching," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, June 2009, pp. 4387–4393.

[31] J. Fink and V. Kumar, "Online methods for radio signal mapping with mobile robots," in *International Conference on Robotics and Automation*, Anchorage, 2010, pp. 1940–1945.

[32] M. Andriluka, M. Friedmann, S. Kohlbrecher, J. Meyer, K. Petersen, C. Reinl, P. Schauß, P. Schnitzspan, D. Thomas, A. Vatcheva, and O. V. Stryk, "Robocuprescue 2009- robot league team darmstadt rescue robot team (germany)," 2009.

[33] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Consistent depth maps recovery from a video sequence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 6, pp. 974–988, june 2009.

[34] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 3, Hong Kong, 0-0 2006, pp. 15–18.

[35] H. Sunyoto, W. van der Mark, and D. Gavrila, "A comparative study of fast dense stereo vision algorithms," in *Intelligent Vehicles Symposium, 2004 IEEE*, Parma, june 2004, pp. 319–324.

[36] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[37] K. Klasing, D. Wollherr, and M. Buss, "A clustering method for efficient segmentation of 3D laser data," in *International Conference on Robotics and Automation*, Pasadena, 2008.

[38] C. Rasmussen, "Combining laser range, color, and texture cues for autonomous road following," in *International Conference on Robotics and Automation*, Washington, D.C., 2002.

[39] S. Gächter, A. Harati, and R. Siegwart, "Structure verification toward object classification using a range camera," in *Conference on Intelligent and Autonomous Systems*, vol. 10, 2008.

[40] L. Spinello, R. Triebel, and R. Siegwart, "Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction," in *International Conference on Robotics and Automation*, Pasadena, 2008.

[41] Q. Zhan, Y. Liangb, and Y. Xiaoa, "Color-based segmentation of point clouds," in *Laser Scanning 2009*, vol. IAPRS XXXVIII Part 3/W8, Paris, Sept. 2009.

[42] K. Klasing, D. Wollherr, and M. Buss, "Realtime segmentation of range data using continuous nearest neighbors," in *International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[43] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[44] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, nov 2000.

[45] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, San Juan, Puerto Rico, jun 1997, pp. 1106–1112.

[46] S. K. Vaios Balis, I. KOTSIS, C. Liapakis, and N. Simpas, "3D - laser scanning: Integration of point cloud and CCD camera video data for the production of high resolution and precision rgb textured models: Archaeological monuments surveying application in ancient Ilida," in *Virtual Systems and Multimedia*, 2004.

[47] H. Aliakbarpour, P. Núñez, J. Prado, K. Khoshhal, and J. Dias, "An efficient algorithm for extrinsic calibration between a 3D laser range finder and a stereo camera for surveillance," in *ICAR 2009*, Munich, Jun. 2009.

[48] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," in *Intelligent Robots and Systems*, San Deigo, November 2007, pp. 4164–4169.

[49] J. Wassenberg, W. Middelmann, and P. Sanders, "An efficient parallel algorithm for graph-based image segmentation," in *CAIP '09: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns'*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 1003–1010.

[50] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, May 2011.

[51] O. P. Frank Moosmann and C. Stiller, "Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion," in *Intelligent Vehicles Symposium*, 2009.

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2001, ch. 21.

[53] C. Erdogan, M. Paluri, and F. Dellaert, "Planar segmentation of RGBD images using fast linear fitting and markov chain monte carlo," in *Computer and Robot Vision (CRV), 2012 Ninth Conference on*. Toronto: IEEE, 2012, pp. 32–39.

[54] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D lidar point clouds," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. Shanghai: IEEE, 2011, pp. 2798–2805.

[55] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3D lidar and camera by maximizing mutual information," *Proceedings of the AAAI National Conference on Artifical Intelligence*, 2012.

[56] J. Strom, A. Richardson, and E. Olson, "Graph-based segmentation for colored 3D laser point clouds," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Taipei, October 2010.

[57] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: A system for large-scale mapping in constant-time using stereo," *International Journal of Computer Vision*, pp. 1–17, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11263-010-0361-7

[58] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time," in *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[59] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. San Francisco: IEEE, 2010, pp. 1498–1505.

[60] J.-Y. Bouguet, "Camera calibration toolbox for MATLAB," July 2010.

[61] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[62] J. Barreto, J. Roquette, P. Sturm, F. Fonseca, *et al.*, "Automatic camera calibration applied to medical endoscopy," in *20th British Machine Vision Conference (BMVC'09)*, London, 2009.

[63] A. R. Jensenius, K. Nymoen, S. A. v. D. Skogstad, and A. Voldsund, "A study of the noise-level in two infrared marker-based motion capture systems," pp. 258–263, 2012.

[64] M. Li, "Camera calibration of a head-eye system for active vision," *Computer VisionECCV'94*, pp. 541–554, 1994.

[65] J. Barreto and K. Daniilidis, "Wide area multiple camera calibration and estimation of radial distortion," in *Omnivis-2004, ECCV-2004 workshop*, Prague, 2004.

[66] T. Svoboda, "A software for complete calibration of multicamera systems," pp. 115–128, 2005.

[67] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A general framework for graph optimization," in *ICRA*, Shanghai, May 2011.

[68] K. Konolige, "Sparse sparse bundle adjustment," in *British Machine Vision Conference*, Aberystwyth, Wales, August 2010.

[69] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[70] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 8, pp. 1335–1340, 2006.

[71] A. Richardson, J. Strom, and E. Olson, "AprilCal: Assisted and repeatable camera calibration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, November 2013.

[72] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, Kobe, Japan, 2009.

[73] V. Pradeep, K. Konolige, and E. Berger, "Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach," in *International Symposium on Experimental Robotics (ISER)*, New Delhi, India, 12/2010 2010.

[74] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter, "Visually mapping the rms titanic: Conservative covariance estimates for slam information filters," *The International Journal of Robotics Research*, vol. 25, no. 12, p. 1230, 2006.

[75] R. M. Eustice, "Large-area visually augmented navigation for autonomous underwater vehicles," Ph.D. dissertation, Massachusetts Institute of Technology, 2005.

[76] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller, "An Atlas framework for scalable mapping," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1899–1906.

[77] J. Strom and E. Olson, "Communication optimization with mutual modeling (COMM): Reducing communication requirements for multi-robot teams," Tech. Rep., 2015.

[78] R. Goeddel and E. Olson, "Inferring categories to accelerate the learning of new classes," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[79] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, June 2015.